Dasar-Dasar Pemrograman 2

Lab 2 Kelas C & F

Java Fundamentals :
Data Types (Strings and Characters), Conditionals, Loops, and Methods

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

Selasa, 19 Feb 2019 - 16:00 WIB

I. Data Types : Strings and Characters

Strings

String dapat diartikan sebagai kumpulan karakter. Pada Java, string merupakan salah satu *reference data type* (akan dijelaskan lebih lanjut di sesi lab). Seperti pada Python, string di Java juga immutable. Berikut beberapa method yang dimiliki oleh String objects:

TABLE 4.7 Simple Methods for `String` Objects

Method	*Description*
`length()`	Returns the number of characters in this string.
`charAt(index)`	Returns the character at the specified index from this string.
`concat(s1)`	Returns a new string that concatenates this string with string s1.
`toUpperCase()`	Returns a new string with all letters in uppercase.
`toLowerCase()`	Returns a new string with all letters in lowercase

(*Source*: Introduction to Java Programming, Comprehensive Version, 10th Edition. Daniel Y. Lang)

Tidak seperti di Python, perbandingan antara 2 string tidak bisa menggunakan operasi pembanding matematis (>, >=, <, <=). Berikut method yang disediakan :

TABLE 4.8 Comparison Methods for `String` Objects

Method	*Description*
`equals(s1)`	Returns true if this string is equal to string s1.
`equalsIgnoreCase(s1)`	Returns true if this string is equal to string s1; it is case insensitive.
`compareTo(s1)`	Returns an integer greater than 0, equal to 0, or less than 0 to indicate whether this string is greater than, equal to, or less than s1.
`compareToIgnoreCase(s1)`	Same as `compareTo` except that the comparison is case insensitive.
`startsWith(prefix)`	Returns true if this string starts with the specified prefix.
`endsWith(suffix)`	Returns true if this string ends with the specified suffix.

(*Source*: Introduction to Java Programming, Comprehensive Version, 10th Edition. Daniel Y. Lang)

Contoh :

```
String example = "example string";
System.out.println(example.length());                          // 14
System.out.println(example.charAt(0));                         // 'T'
System.out.println(example.equals(new String("example string")));      // true
```

Notes : Ingat, String literal di java diapit oleh double quote (“)

Chars

Mirip seperti String, hanya saja char merupakan *primitive data type* dan char hanya terdiri dari 1 karakter saja. **Jika string diapit oleh double quote (“), char diapit oleh single quote (‘).**

Char memiliki beberapa method:

`isDigit(ch)`	Returns true if the specified character is a digit.
`isLetter(ch)`	Returns true if the specified character is a letter.
`isLetterOfDigit(ch)`	Returns true if the specified character is a letter or digit.
`isLowerCase(ch)`	Returns true if the specified character is a lowercase letter.
`isUpperCase(ch)`	Returns true if the specified character is an uppercase letter.
`toLowerCase(ch)`	Returns the lowercase of the specified character.
`toUpperCase(ch)`	Returns the uppercase of the specified character.

(*Source*: Introduction to Java Programming, Comprehensive Version, 10th Edition. Daniel Y. Lang)

Meski char menyimpan 1 karakter, beberapa karakter disimbolkan dengan lebih dari 1 karakter, seperti :

TABLE 4.5 Escape Sequences

Escape Sequence	*Name*
`\b`	Backspace
`\t`	Tab
`\n`	Linefeed
`\f`	Formfeed
`\r`	Carriage Return
`\\`	Backslash
`\"`	Double Quote

(*Source*: Introduction to Java Programming, Comprehensive Version, 10th Edition. Daniel Y. Lang)

Tidak seperti String, karena char merupakan *primitive data type*, maka char dapat dibandingkan dengan operasi > , >=, <, <=, ==, dan !=. Ingat, yang dibandingkan adalah kode ASCII dari char tersebut. Anda telah mempelajari nya di DDP 1.

II. Conditionals

Serupa dengan conditionals yang dahulu dipelajari di DDP 1, conditionals di Java tidak jauh berbeda, hanya perbedaan sintaks saja. Perhatikan contoh berikut :

If - Else If - Else

```
int age = 18;

if (age < 2) {
      System.out.println("You are a baby");
} else if (age < 15) {
      System.out.println("Still a kid to me");
} else {
      System.out.println("Cool teenager");
}

// prints "Cool teenager"
```

Switch Case

Di Java, juga ada switch yang serupa dengan rentetan conditionals :

```
switch(command.toUpperCase()) {
      case "HELP" : System.out.println("Here is the command menus : .."); break;
      case "EXIT" : System.out.println("Exiting program now.."); break;
      case "START" : System.out.println("Program is now running.."); break;
      case "STOP" : System.out.println("Program stopped");
}
```

Ternary Operator

Kemudian, Java juga mendukung operasi ternary yang memiliki aspek conditional juga, seperti :

```
Scanner scanner = new Scanner(System.in);
int input = scanner.nextInt();

int odd = input % 2 == 0 ? false : true;
```

Assignment pada variable *odd* di atas serupa dengan :

```
int odd;
if (input % 2 == 0) {
      odd = false;
} else {
      odd = true;
)
```

III. Loops

Di Java, dikenal 3 bentuk loop yang umum :

For-Loop

For-loop biasa digunakan ketika jumlah perulangan yang akan dilakukan sudah diketahui sejak awal.

```
for (start; condition; mutation) {
      // statements
}
```

Start merupakan kondisi awal, biasa di *assign* nilai awal sebuah counter variable.
Condition merupakan kondisi dimana jika menghasilkan nilai true, statements dalam block dijalankan (lagi). Pengecekan dilakukan SEBELUM menjalankan block statements.
Mutation merupakan pengubah. Pengubahan dilakukan SETELAH block statements selesai dijalankan. Biasa digunakan untuk mengubah counter variable.

Contoh :

```
For (int i = 0; i < 5; i++) {
      System.out.print(i);
}

// prints 01234
```

While-Loop

Perulangan dengan while biasa dilakukan ketika jumlah perulangan tidak diketahui sejak awal.

```
while (condition) {
      // statements
}
```

Condition merupakan kondisi dimana jika menghasilkan nilai true, maka statements akan dijalankan. Pengecekan dilakukan SEBELUM memasuki block statements.

Contoh:

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

while(!scanner.nextLine().equals("exit")) {
      System.out.println("type <exit> to quit this program");
}

// prints "type <exit> to quit this program" until user types "exit"
```

Do-While-Loop

Serupa dengan while-loop, hanya saja pengecekan dilakukan SETELAH block statements dijalankan. Jadi block statements pasti dilakukan minimal 1 kali.

```
do {
      // statements
} while (condition);
```

Condition merupakan kondisi dimana jika menghasilkan nilai true, block statements akan dijalankan. Perhatikan bahwa pengecekan dilakukan SETELAH block statements dijalankan.
Contoh :

```
int x = 3;
do {
      System.out.println("statement run");
} while (x == 0);

// prints "statement run" 1 time
```

Untuk ketiga bentuk loop ini, terdapat **loop-exiting command** yang fungsinya sudah kalian kenali : **break** (keluar dari loop) dan **continue** (*skip* 1 iterasi).

IV. Math Library and Functions

Library Math sudah terimport secara default oleh Java. Function yang disediakan dapat membantu kita dalam menjalankan operasi-operasi matematis. Berikut beberapa function yang disediakan :

TABLE 4.2 Exponent Methods in the Math Class

Method	*Description*
`exp(x)`	Returns e raised to power of x (e^x).
`log(x)`	Returns the natural logarithm of x ($\ln(x) = \log_e(x)$).
`log10(x)`	Returns the base 10 logarithm of x ($\log_{10}(x)$).
`pow(a, b)`	Returns a raised to the power of b (a^b).
`sqrt(x)`	Returns the square root of x ($\sqrt{x}$) for x >= 0.

(*Source*: Introduction to Java Programming, Comprehensive Version, 10th Edition. Daniel Y. Lang)

Pemanggilan menggunakan prefix Math, contoh : Math.sqrt(number)

Selain itu, ada constant seperti **Math.PI** dan operasi-operasi yang lebih umum digunakan seperti:

```
Math.max(2, 3) returns 3
Math.max(2.5, 3) returns 4.0
Math.min(2.5, 4.6) returns 2.5
Math.abs(-2) returns 2
Math.abs(-2.1) returns 2.1
```

(*Source*: Introduction to Java Programming, Comprehensive Version, 10th Edition. Daniel Y. Lang)

Selengkapnya dapat di lihat di : https://docs.oracle.com/javase/10/docs/api/java/lang/Math.html.

V. Methods

Pada DDP 1 kalian telah mengenal konsep functions and methods - sebuah *mini* program (block of code) yang ketika dipanggil akan menjalankan perintah-perintah yang didefinisikan di dalamnya. Berikut adalah format **umum** pendefinisian suatu method di Java.

```
accessModifier returnValueType methodName(parameterType1 parameterName1, ...) {
      // statements
}
```

accessModifier menentukan siapa-siapa saja yang dapat mengakses method ini. Untuk sekarang, kita akan menggunakan modifier **public** yang berarti semua instance/class dapat mengaksesnya.
returnValueType menentukan tipe data yang di *return* oleh function ini. Jika function nya tidak me *return* apa-apa, gunakan keyword **void**.
methodName dan **parameters** menjelaskan bagaimana function ini dipanggil dan apa saja parameter yang dibutuhkan.

public static void main (String[] args) adalah salah satunya, berikut contoh lainnya :

```
public static int multiply(int a, int b) {
      return a * b;
}

int a = 2;
int b = 3;
int c = multiply(a,b);

// c = 6
```

Notes : **static** adalah *keyword* yang menandakan bahwa method ini tidak dimiliki oleh sebuah instance class tertentu sehingga method ini bisa dipanggil oleh siapa saja.

Method Overloading

Java mengenal konsep method overloading (bedakan dengan *overriding*). Overloading merupakan konsep dimana terdapat 2 atau lebih method dengan **nama** yang sama, tetapi parameter(s) nya berbeda : entah itu jumlahnya, tipenya, atau urutan tipenya. **Java akan secara otomatis memilih method yang sesuai dengan parameter yang diberikan**. Berikut contohnya:

```
public static int multiply(int a, int b) {
      return a * b;
}

public static double multiply(double a, double b) {
      return a * b;
}
```

[Tutorial] Air Mancur Taman Firdaus

Baru-baru ini ayah Dek Depe, seorang pengusaha kopi Jawa, hendak menghias tamannya: Taman Firdaus, dengan membangun suatu air mancur baru berbentuk lingkaran yang memiliki tulisan terukir di sisi nya. Beliau meminta Dek Depe untuk membuat program yang bisa membantu proses pembangunannya. Namun Dek Depe kembali menemukan kesulitan dalam memilih *data type* yang harus digunakan dalam kode, mengimplementasikan *conditional* dan *loop* dalam kode, memanfaatkan *Math library* and *functions*, dan membungkus semua kode tersebut dalam sebuah *method*. Karena kamu adalah programmer yang handal, baik, dan suka menolong, Dek Depe yakin kamu pasti ingin membantu dia dengan setulus hati. (*image source* : depositphotos.com)

Kamu diminta membuat sebuah program Java PenghitungAirMancur.java yang bisa menghitung lebar lahan (diameter air mancur) *minimal* yang diperlukan untuk membangun air mancur lingkaran dengan tulisan di sisinya dengan spesifikasi sebagai berikut:

Spesifikasi Program :

Main Method:

1. Program akan menerima input sebuah String.
2. String ini diukir dalam ubin-ubin keramik (*tiles*) dan disusun mengelilingi air mancur. Program akan menghitung banyak ubin yang dibutuhkan untuk keseluruhan String tersebut, lalu menghitung lebar minimal yang dibutuhkan untuk air mancur tersebut dalam satuan *tiles*. (Membutuhkan bantuan dari method lain)
3. Cetak output berupa lebar (diameter) minimal air mancur tersebut dalam satuan *tiles*.

Method *hitungKeliling* (yang menentukan banyaknya ubin yang diperlukan untuk mengukir String mengelilingi air mancur):

1. Iterasi setiap karakter yang ada pada String tersebut.
2. Jumlahkan jumlah ubin yang diperlukan untuk tiap karakter.
    a. Huruf alphabet kapital membutuhkan 3 *tiles*.
    b. Huruf alphabet non-kapital membutuhkan 1 *tile*.
    c. Karakter lain membutuhkan 2 *tiles*.

Method *hitungLebar* (yang menentukan lebar air mancur minimal):

1. Gunakan formula **diameter = keliling/π.**
2. Hasil harus dalam bentuk integer (jumlah *tiles* tidak dapat dalam bentuk pecahan). Ingat, yang diminta adalah jumlah minimal, jadi jika penghitungan menghasilkan 7.5 *tiles*, maka jumlah yang dibutuhkan adalah 8 *tiles*. Bagaimana cara *rounding double* lalu diubah ke integer? Sepertinya module Math memiliki function khusus untuk *rounding double* :)

Contoh Input/Output :

```
Masukkan teks yang akan diukir : Fasilkom UI
Anda membutuhkan lebar lahan minimal : 6 tiles
```

Notes : karakter bercetak tebal-miring merupakan input dari user

Penjelasan Contoh :

1. String yang diinput oleh user adalah “Fasilkom UI”
2. Mengikuti ketentuan yang diberikan, maka banyak *tiles* untuk menyusun String ini adalah :
   3 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 2 + 3 + 3 = 18 *tiles*
3. Maka, lebar yang dibutuhkan adalah
   18 / π = 5.729577951308232 *tiles*
4. Sehingga, lebar minimum yang diperlukan adalah 6 tiles (pembulatan ke atas).

Komponen Penilaian :

Komponen	Penjelasan	Bobot
Implementasi fungsi *main*	Mengimplementasikan fungsi *main* dengan benar dalam hal membaca input, memanggil fungsi-fungsi lain, dan mengeluarkan output.	25 %
Penggunaan tipe data yang benar.	Menggunakan tipe data yang benar pada variabel-variabel dan juga output setiap fungsi.	10%
Penggunaan *library* Math	Menggunakan class Math dengan benar.	5%
Implementasi fungsi *hitungKeliling*	Mengimplementasikan fungsi *hitungKeliling* sesuai dengan algoritma di atas dan mengeluarkan output yang sesuai.	35%
Implementasi fungsi *hitungLebar*	Mengimplementasikan fungsi *hitungKeliling* agar mengeluarkan output yang sesuai.	10%
Kerapian	Penulisan program mengikuti kaidah dan konvensi yang telah diajarkan. Program ditulis dengan rapi, dan terstruktur	15 %

Deadline :

Selasa, 19 Feb 2019
Pukul **17:20 WIB**

Format Pengumpulan :

Kumpulkan di slot pengumpulan yang telah disediakan di SCeLE dengan format :

PenghitungAirMancur.Java yang akan dicompress ke dalam zip dengan format:

[Kode Asdos]_[Nama]_[Kelas]_[NPM]_Lab2.zip

Contoh :

JO_JonathanChristopherJakub_C_1706040151_Lab2.zip

Acknowledged Lecturers :

- Laksmita Rahadianti, S.Kom., M.Sc., Ph.D.
- Fariz Darari, S.Kom, M.Sc., Ph.D.

Authors :

- AN
- DR
- JO
- SAM
- KC