Dasar-Dasar Pemrograman 2

Lab 02

Recursion

I. 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.

Contoh paling umum yang paling sering kita temukan:

```
public static void main(String[] args) {}
```

Selain contoh di atas, 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
```

Note: **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.

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

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

II. Recursion

Pada tutorial kali ini, Anda akan belajar suatu konsep di dalam pemrograman yang disebut dengan ***recursive method***. Sebuah method/fungsi dapat dikatakan rekursif jika fungsi tersebut dapat memanggil (meng-*invoke*) dirinya sendiri baik secara langsung maupun tidak langsung. Biasanya, fungsi rekursif digunakan untuk menyelesaikan permasalahan-permasalahan yang dapat direduksi menjadi masalah serupa yang lebih kecil. Selain itu, penggunaan teknik rekursi juga dapat membuat kode menjadi lebih ringkas, elegan, dan simpel, ketika digunakan dengan baik.

Dalam fungsi rekursif, terdapat dua komponen utama yang saling berkaitan:

- ***Base case*** (kasus dasar atau kasus penyetop)
  Base case merupakan permasalahan terkecil yang bisa diselesaikan dengan mudah tanpa melibatkan pemecahan masalah menjadi masalah yang lebih kecil lagi. Dalam fungsi rekursif, base case tidak melakukan sebuah pemanggilan rekursif lagi (recursive call), melainkan bertujuan untuk **memberhentikan proses rekursi**.
- ***Recursive case***
  Recursive case yaitu sebuah proses yang melakukan pemecahan permasalahan lebih **besar** menjadi permasalahan **kecil** yang **identik**. Pastikan proses ini akan mengarah kepada *base case*.

Supaya kamu mendapat gambaran lebih jelas mengenai rekursi, perhatikan contoh fungsi perhitungan faktorial sebuah bilangan di bawah ini.

```
public static int factorial(int number) {
  if (number <= 1) {
    return 1; // Base case
  } else {
    return number * factorial(number - 1); // Recursive case
  }
}
```

Pada fungsi di atas, *base case* dari fungsi tersebut adalah ketika angka pada parameter **number** kurang dari atau sama dengan 1 (1! = 1 dan 0! = 1, diasumsikan juga **number** tidak negatif). *Base case* ini tidak melakukan *recursive call* sama sekali karena bertujuan untuk menghentikan proses rekursi.

Kemudian, proses rekursif pada fungsi di atas akan melakukan komputasi dengan mengembalikan nilai **number** itu sendiri dan dikalikan dengan pemanggilan kembali fungsi **factorial** dengan parameter yang berubah (perhatikan bahwa parameter mengecil dan mengarah ke *base case*).

Misalnya, pemanggilan **factorial(3)** akan mengembalikan nilai 3, kemudian dikali dengan **factorial(2)** yang akan mengembalikan nilai 2 dikali dengan **factorial(1)**, dan akhirnya berakhir ke *base case*, proses rekursi berhenti, dan akhirnya fungsi mengembalikan nilai 3 * 2 * 1 = 6.

Untuk diingat: Hati-hati ketika mendefinisikan fungsi rekursif, jangan sampai terjadi rekursi yang tak hingga (*infinite recursion*) yang akan menyebabkan StackOverflowError. Untuk menghindarinya, pastikan ada *base case*, dan setiap langkah rekursif selalu mengarah ke *base case*.

III. String methods

Objek bertipe data String memiliki beberapa *method* yang bisa kamu manfaatkan untuk mengolahnya. *Method* yang dapat berguna dalam pengerjaan soal lab pekan ini di antaranya adalah .substring() dan .charAt().

1. substring()
   Method substring() memiliki dua versi, yakni substring(firstIndex) dan substring(firstIndex, lastIndex). *Method* substring(firstIndex) mengembalikan substring dari indeks firstIndex (inklusif) sampai akhir. Sementara itu, *method* substring(firstIndex, lastIndex) mengembalikan substring dari indeks firstIndex sampai lastIndex-1.
   Contoh:

```
String message = "Hello, world!";
String substr1 = message.substring(3);  // "lo, world!"
String substr2 = message.substring(2, 9);  // "llo, wo"
```

2. charAt()
   Method charAt(i) mengembalikan karakter (bertipe data `char`) pada index ke-i dalam string.

```
String message = "Hello, world!";
char c = message.charAt(1);  // 'e'
```

Selengkapnya, silakan buka dokumentasi Java mengenai String.

Soal Lab 02

LET'S GET OUT OF THIS PLANET

Source: https://www.google.com

Dikisahkan pada suatu waktu, Bumi sudah menjadi tempat yang tidak layak untuk dihuni oleh seluruh makhluk hidup di dalamnya. Tidak hanya jumlah populasi manusia yang sudah membeludak, tetapi juga tingkah keseharian manusia yang terus menerus merusak Bumi. Pada akhirnya, Bumi pun melaksanakan misi balas dendamnya. Gempa bumi, gunung meletus, badai, banjir, hingga longsor terjadi di berbagai pelosok Bumi. Akibatnya, Salman dan Dek Depe diberikan misi rahasia oleh NASA untuk mencari tempat yang layak untuk dihuni oleh berbagai makhluk hidup di luar tata surya kita.

Namun, mereka berdua tidak sendiri, karena misinya adalah membawa sebanyak-banyaknya makhluk hidup ke planet lain. Salman membawa Meong untuk ikut keluar angkasa sebagai representasi kecocokan hewan di planet yang baru nanti. Sementara itu, Dek Depe ditugaskan untuk tetap di Bumi sembari memonitori kinerja Salman dan Meong.

Meong dan Salman pun bersiap-siap, kemudian berangkatlah mereka dengan pesawat luar angkasa. Setelah mereka berhasil melewati seluruh lapis atmosfer Bumi, mereka pun mulai menjelajahi planet satu per satu. Jam demi jam berhasil dilewati oleh mereka berdua, tetapi belum ada satu pun yang cocok dengan planet yang dibutuhkan, hingga pada akhirnya Salman pun merasa *awkward* dengan Meong karena sepanjang perjalanan tadi, Salman belum kenalan dengan Meong. Namun, ketika hendak berkenalan Salman lupa kalau dia tidak bisa berbahasa kucing. Akhirnya Salman meminta Dek Depe untuk membuat program penerjemah bahasa kucing menjadi bahasa yang mudah dimengerti. Namun, Dek Depe kesulitan dalam menyelesaikan program ini.

Sebagai teman Dek Depe yang baik sekaligus ingin menjadi pahlawan yang menyelamatkan bangsa manusia, kamu akan membantu Dek Depe menerjemahkan bahasa kucing agar mudah dimengerti oleh Salman selama menjalani misi ini dengan menemukan pola dari setiap huruf yang kucing ucapkan dengan membuat program menggunakan Java.

Soal Lab:

Meong berkomunikasi ke Salman dengan berbicara dan dengan gesturnya. Salman akan menerima:

1. **Nada di setiap hurufnya**
   Program akan menerima input String yang sulit terbaca. Input ini merupakan sebuah kata yang telah diberikan variasi nada sebesar x. Variasi nada yang diterima beragam antara 1, 2, 3, 4, 5, 6, dan 7. Setiap nada yang dikeluarkan oleh kucing berarti 1 huruf yang dimaksud akan naik sebanyak x nada ascii. Sebagai contoh:
   - “zxzl” dengan nada 1111 menghasilkan “ayam”
   - “xdapx” dengan nada 21123 menghasilkan “zebra”

   Note:
   - Asumsikan semua input pasti *lowercase* dan variasi nada yang diterima pasti ada di antara 1, 2, 3, 4, 5, 6, dan 7. Kamu tidak perlu memeriksa kedua hal tersebut di dalam kodemu.
   - Jumlah huruf dalam suatu kata dipastikan >= 1, tetapi tidak memiliki batasan atas yang ditentukan.
   - Jika 'z' diberi 1 nada, maka akan kembali ke 'a'.
   - Jumlah panjang input nada dipastikan sesuai dengan panjang input kata (tidak perlu diperiksa).

2. **Gestur**
   Setiap kali terakhir Meong mengucapkan katanya, Meong memberikan gestur berupa tepukan sebanyak n. Bilangan n menunjukkan banyaknya pengulangan kata. Sebagai contoh:
   - Setelah input “zxzl” dan 1111. Tepukan = 2
     Hasilnya adalah “ayamayam”

Untuk menghilangkan keraguan dalam mengerjakan soal, berikut contoh-contoh kasus yang mungkin ditemui untuk program ini sebagai berikut:

INPUT	**OUTPUT**
Masukkan kata: jtot Masukkan nada: 1111 Masukkan jumlah tepukan: 2	**kupukupu**
Masukkan kata: bhmsz Masukkan nada: 11111 Masukkan jumlah tepukan: 1	**cinta**

Masukkan kata: kysle Masukkan nada: 22222 Masukkan jumlah tepukan: 1	**maung**
Masukkan kata: gxjl Masukkan nada: 1323 Masukkan jumlah tepukan: 3	**halohalohalo**
Masukkan kata: aaaaa Masukkan nada: 54321 Masukkan jumlah tepukan: 2	**fedcbfedcb**
Masukkan kata: jcqccccccccccm Masukkan nada: 121222222222221 Masukkan jumlah tepukan: 1	**kereeeeeeeeeeen**

Catatan:

1. Jangan lupa untuk berdoa sebelum mengerjakan, berusahalah untuk tenang dan jangan panik.
2. Baca dan pahami materi dan soal dengan baik.
3. Sangat disarankan untuk merancang solusimu terlebih dahulu sebelum menulis kode program. Dengan begitu, kamu dapat lebih memahami alur solusimu dan meminimalkan *error* pada program.
4. Buat program tersebut menggunakan Java dengan mengimplementasikan kedua proses secara **REKURSIF**. Gunakan pengetahuanmu tentang fungsi pada Java untuk membuat kode rekursif. Akan ada **pengurangan nilai yang signifikan** jika kamu tidak menerapkan rekursi dalam solusimu.
5. Tersedia *template* untuk memudahkan kamu dalam mengerjakan soal, tetapi *template* tidak wajib kamu gunakan. Jika *template* malah membuat kamu bingung, lebih baik buatlah kode sendiri dari awal.
6. Hint: Gunakan *method* .substring() untuk melakukan *string slicing* (seperti [:] di Python). Kamu juga bisa menggunakan *method* .charAt() apabila diperlukan, tetapi harap diperhatikan bahwa .charAt() hanya mengembalikan satu karakter dalam tipe data char (bukan String).
7. Jangan ragu untuk bertanya kepada asisten dosen apabila mengalami kesulitan atau menemukan ketidakjelasan pada soal.
8. Terdapat komponen penilaian sebesar **10%** untuk kerapian kode dan dokumentasi.

Selamat Mengerjakan,
Good Luck!