Dasar-Dasar Pemrograman 2

Tutorial 4 Kelas B, D, & E

Recursion

Jumat, 8 Maret 2019 - 10:00 WIB

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)**
  Yaitu 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***
  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.

```
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*.

Toko Permen

Setelah bersenang-senang dengan Dek Depi bermain catur, Dek Depe menjadi lelah karena terlalu keras berpikir bagaimana cara kuda bergerak di papan catur. Karena Dek Depi memiliki hobi makan, dia memutuskan untuk mencari makan di toko-toko yang tersedia dekat tempat Dek Depi bermain catur. Dek Depi sangat suka makanan yang manis (karena doi suka yang manis-manis), dan toko yang terdekat adalah Toko Permen.

Toko Permen memberikan tantangan kepada Dek Depi, di mana jika Dek Depi berhasil menyelesaikannya, Dek Depi bisa mendapat permen gratis. Pemilik toko memberikan sebuah daftar kode (arrayOfItemCodes) dan daftar permen yang dijual (arrayOfItems). Kode pada arrayOfItemCodes adalah hasil hash dari permen yang bersesuaian. Tantangannya adalah Dek Depi harus mengidentifikasi permen apa saja (sesuai urutan di daftar kode) yang sebenarnya ada di daftar kode itu. Ada dua buah method yang harus kalian buat, yaitu binarySearch dan hash.

binarySearch

1. Pada array terurut yang disediakan, cari elemen tengah nya dan bandingkan nilai **hash** elemen tersebut dengan kode yang anda cari (selanjutnya akan disebut dengan **searched**).
2. Jika **searched** ternyata lebih besar daripada hash elemen tengah, pastilah **searched** berada di array sisi kanan (dari tengah sampai akhir kanan) bukan? Dan jika **searched** ternyata lebih kecil daripada hash elemen tengah, pastilah ia berada di array sisi kiri. Jika **searched** sama dengan hash elemen tengah, return elemen tengah ini karena kita telah menemukannya. Asumsi kita saat ini adalah elemen nya ADA di dalam array.
3. Apabila elemen belum ditemukan, pencarian berikutnya dapat mencari di setengah array sebelumnya saja (sisi kanan atau sisi kiri nya).
4. Jika array pencarian sudah tidak bisa dibagi lagi dan kita masih belum menemukan **searched**, berarti **searched** tidak ada dalam array.

hash

1. Hash function yang sederhana ini didefinisikan dengan

$$\sum_{i=0}^{str.length()} int(str[i]) \times (i+1)$$

2. Catatan : Karakter disini *case-insensitive* dan untuk karakter 'A' dimulai dari 1 dan seterusnya. Karakter yang dihitung hanyalah karakter yang Alfabetik, tetapi *index* nya tetap mengikuti *indexing* pada string. (misal "A B" berarti jumlahkan 1x1 dan 2x3, menghasilkan hash bernilai 7). Int(str[i]) pada formula bernilai posisi suatu karakter dalam alphabet, sehingga pada contoh "A B", A bernilai 1, B bernilai 2, begitu seterusnya hingga karakter Z bernilai 26.
3. Buat fungsi rekursif untuk mengimplementasikan fungsi ini.

Kalian akan diberikan *template* agar memudahkan kalian mengerjakan tantangan ini. *Template* berisi method main yang sudah lengkap, method driver binarySearch dan hash yang tidak perlu kalian ubah (jika kalian mau, silahkan, baca notes di atas template yang disediakan), dan method rekursif binarySearch dan hash yang harus kalian lengkapi. Cukup lengkapi bagian yang memiliki "TODO" pada *template*.

Contoh Output:

```
Found item : Chocolate Bar [ 50.00 ]
Found item : Random Taste Candy [ 7.15 ]
Found item : Marshmallow [ 12.75 ]
Found item : Lollipop [ 1.80 ]
Found item : Chocolate Bar [ 50.00 ]
Found item : Poprocks [ 3.45 ]
No item found
```

Pemeriksa dapat mengubah urutan atau isi dari arrayOfItemCodes untuk memastikan implementasi kalian benar-benar sesuai, tidak hanya untuk kombinasi daftar kode hash yang ada di *template* saja.

Komponen Penilaian :

Komponen	Penjelasan	Bobot
Tampilan Program	Interaksi program yang dibuat sesuai dengan contoh.	10%
Implementasi Fungsi hash	Mengimplementasikan fungsi hash seperti deskripsi soal dengan **menggunakan rekursi**.	25%
Implementasi Fungsi binarySearch	Mengimplementasikan fungsi binarySearch seperti deskripsi soal dengan **menggunakan rekursi**.	25%
Kebenaran Program	Kesesuaian Output program dengan hasil yang diinginkan	25%
Kerapian Kode	Penulisan program mengikuti kaidah dan konvensi yang telah diajarkan. Program ditulis dengan rapi, terstruktur, dan disertakan oleh dokumentasi secukupnya	15 %

Deadline : Jumat, 8 Maret 2019

Pukul **23:55 WIB**

Format Pengumpulan :

Program dibuat dalam sebuah file Java yang bernama **CandyStore.java**

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

[Kode Asdos]_[Nama]_[Kelas]_[NPM]_Lab4.zip yang berisi **CandyStore.java** saja

Contoh :

TES_DemoSuremo_C_1234567891_Lab4.zip yang berisi **CandyStore.java** di dalamnya.

Acknowledged Lecturers :

- Fariz Darari, S.Kom, M.Sc., Ph.D.
- Dipta Tanaya, S.Kom., M.Kom.

Authors :

- JO
- KC
- SAM

Selamat Bekerja!