Rekursive

Dari Kelas kita sebelumnya di Java, kita telah melihat pendekatan iteratif dimana kita mendeklarasikan sebuah loop dan kemudian melintasi struktur data secara iteratif dengan mengambil satu elemen pada satu waktu.

Kita juga telah melihat aliran bersyarat di mana lagi kita menyimpan satu variabel loop dan mengulangi sepotong kode sampai variabel loop memenuhi kondisi. Dalam hal pemanggilan fungsi, kita juga mempelajari pendekatan berulang untuk pemanggilan fungsi.

- Apa Rekursive Di Jawa?
  - Memahami Rekursive Di Jawa
    - Kondisi Basis Rekursive
    - Pemecahan Masalah Menggunakan Rekursive
    - Stack Overflow Error Dalam Rekursive
  - Contoh Rekursive Di Jawa
    - # 1) Seri Fibonacci Menggunakan Rekursive
    - # 2) Periksa Apakah Sebuah Nomor Adalah Palindrome Menggunakan Rekursive
    - # 3) Reverse String Recursion Java
    - # 4) Pencarian Biner Rekursive Java
    - # 5) Temukan Nilai Minimum Dalam Array Menggunakan Rekursive
  - Jenis Rekursive
    - # 1) Rekursive Ekor
    - # 2) Rekursive Kepala
- Rekursive Vs Iterasi Di Jawa

- Pertanyaan yang Sering Diajukan
- Kesimpulan
  - Bacaan yang Direkomendasikan
-

Apa Rekursive Di Jawa?

Rekursive adalah proses di mana fungsi atau metode memanggil dirinya sendiri berulang kali. Fungsi ini yang dipanggil berulang kali baik secara langsung maupun tidak langsung disebut "fungsi rekursif".

Kita akan melihat berbagai contoh untuk memahami Rekursive. Sekarang mari kita lihat sintaks Rekursive.

Sintaks Rekursive

Setiap metode yang mengimplementasikan Rekursive memiliki dua bagian dasar:

1. Pemanggilan metode yang dapat menyebut dirinya sendiri yaitu rekursif
2. Prasyarat yang akan menghentikan Rekursive.

Perhatikan bahwa prasyarat diperlukan untuk setiap metode rekursif karena, jika kita tidak menghentikan Rekursive maka itu akan terus berjalan tanpa batas dan mengakibatkan tumpukan melimpah.

Sintaks umum rekursi adalah sebagai berikut:

```
methodName (T parameters…)
{
    if (precondition == true)

//precondition or base condition
    {
        return result;
    }
    return methodName (T parameters…);

        //recursive call
}
```

Perhatikan bahwa prasyarat juga disebut kondisi dasar. Kita akan membahas lebih lanjut tentang kondisi dasar di bagian selanjutnya.

Memahami Rekursive Di Jawa

Pada bagian ini, kita akan mencoba memahami proses Rekursive dan melihat bagaimana prosesnya. Kita akan belajar tentang kondisi dasar, stack overflow, dan melihat bagaimana masalah tertentu dapat diselesaikan dengan Rekursive dan detail lainnya.

Kondisi Basis Rekursive

Saat menulis program rekursif, pertama-tama kita harus memberikan solusi untuk kasus dasar. Kemudian kita mengungkapkan masalah yang lebih besar dalam bentuk masalah yang lebih kecil.

Sebagai **contoh,** kita dapat mengambil soal klasik menghitung faktorial sebuah bilangan. Diberikan angka n, kita harus mencari faktorial dari n yang dilambangkan dengan n!

Sekarang mari kita implementasikan program untuk menghitung n faktorial (n!) Menggunakan Rekursive.

```
public class Main{
    static int fact(int n)
    {
        if (n == 1)

 // base condition
            return 1;
        else
            return n*fact(n-1);
    }    public static void main(String[] args) {
    int result = fact(10);

     System.out.println("10! = " + result);
    }
}
```

Keluaran

```
10! = 3628800
```

Dalam program ini, kita dapat melihat bahwa kondisi (n <= 1) adalah kondisi dasar dan ketika kondisi ini tercapai, fungsi mengembalikan 1. Bagian lain dari fungsi tersebut adalah pemanggilan rekursif. Tetapi setiap kali metode rekursif dipanggil, n berkurang 1.

Dengan demikian kita dapat menyimpulkan bahwa pada akhirnya nilai n akan menjadi 1 atau kurang dari 1 dan pada titik ini metode akan mengembalikan nilai 1. Kondisi dasar ini akan tercapai dan fungsinya akan berhenti. Perhatikan bahwa nilai n bisa berupa apa saja asalkan memenuhi kondisi dasarnya.

Pemecahan Masalah Menggunakan Rekursive

Ide dasar di balik penggunaan Rekursive adalah untuk mengungkapkan masalah yang lebih besar dalam bentuk masalah yang lebih kecil. Juga, kita perlu menambahkan satu atau lebih kondisi dasar agar kita dapat keluar dari Rekursive.

Ini sudah ditunjukkan dalam contoh faktorial di atas. Dalam program ini, kita menyatakan n faktorial (n!) Dalam bentuk nilai yang lebih kecil dan memiliki kondisi dasar (n <= 1) sehingga ketika n mencapai 1, kita dapat keluar dari metode rekursif.

Stack Overflow Error Dalam Rekursive

Kita menyadari bahwa ketika metode atau fungsi apa pun dipanggil, status fungsi disimpan di stack dan diambil ketika fungsi tersebut kembali. Tumpukan juga digunakan untuk metode rekursif.

Tetapi dalam kasus Rekursive, masalah mungkin terjadi jika kita tidak menentukan kondisi dasar atau ketika kondisi dasar tidak tercapai atau dieksekusi. Jika situasi ini terjadi maka stack overflow mungkin muncul.

Mari kita perhatikan contoh notasi faktorial di bawah ini.

Di sini kita memberikan kondisi dasar yang salah, n == 100.

```
public class Main
{
    static int fact(int n)
    {
        if (n == 100)

 // base condition resulting in stack overflow
            return 1;
        else
            return n*fact(n-1);
    }
public static void main(String[] args) {
    int result = fact(10);

        System.out.println("10! = " + result);
    }
}
```

Jadi ketika n> 100 metode akan mengembalikan 1 tetapi Rekursive tidak akan berhenti. Nilai n akan terus menurun tanpa batas karena tidak ada kondisi lain untuk menghentikannya. Ini akan berlanjut sampai tumpukan meluap.

Kasus lain adalah ketika nilai n <100. Dalam kasus ini, metode juga tidak akan pernah mengeksekusi kondisi dasar dan mengakibatkan stack overflow.

Contoh Rekursive Di Jawa

Di bagian ini, kita akan menerapkan contoh berikut menggunakan Rekursive.

 # 1) Seri Fibonacci Menggunakan Rekursive

Deret Fibonacci diberikan oleh,

1,1,2,3,5,8,13,21,34,55,…

Urutan di atas menunjukkan bahwa elemen saat ini merupakan penjumlahan dari dua elemen sebelumnya. Juga, elemen pertama dalam deret Fibonacci adalah 1.

Jadi secara umum jika n adalah bilangan saat ini, maka itu diberikan oleh penjumlahan dari (n-1) dan (n-2). Karena elemen saat ini diekspresikan dalam elemen sebelumnya, kita dapat mengungkapkan masalah ini menggunakan Rekursive.

Program untuk mengimplementasikan deret Fibonacci diberikan di bawah ini:

```
public class Main
{
    //method to calculate fibonacci series
    static int fibonacci(int n) {
        if (n <= 1) {
            return n;
        }
        return fibonacci(n-1) + fibonacci(n-2);
    }
    public static void main(String[] args) {
        int number = 10;

        //print first 10 numbers of fibonacci series
        System.out.println ("Fibonacci Series: First 10 numbers:");
        for (int i = 1; i <= number; i++)
        {
            System.out.print(fibonacci(i) + " ");
        }

    }
}
```

Keluaran

```
Fibonacci Series: First 10 numbers:

1 1 2 3 5 8 13 21 34 55
```

2) Periksa Apakah Sebuah Nomor Adalah Palindrome Menggunakan Rekursive

Palindrom adalah urutan yang sama jika kita membacanya dari kiri ke kanan atau dari kanan ke kiri.

Diberikan angka 121, kita melihat bahwa ketika kita membacanya dari kiri ke kanan dan dari kanan ke kiri, itu sama. Karenanya nomor 121 adalah palindrom.

Mari kita ambil angka lain, 1242. Jika kita membacanya dari kiri ke kanan, jumlahnya adalah 1242 dan jika dibaca dari kanan ke kiri berbunyi 2421. Jadi ini bukan palindrom.

Kita menerapkan program palindrome dengan membalik digit angka dan secara rekursif membandingkan angka yang diberikan dengan representasi yang dibalik.

Program di bawah ini mengimplementasikan program untuk memeriksa palindrome.

```
import java.io.*;
import java.util.*;

public class Main {
    // check if num contains only one digit
    public static int oneDigit(int num) {
         if ((num >= 0) && (num < 10))
             return 1;
        else
             return 0;
    }
   //palindrome utility function
    public static int isPalindrome_util (int num, int revNum) throws Exception {
            // base condition; return if num=0
        if (num == 0) {
             return revNum;
        } else { //call utility function recursively
             revNum = isPalindrome_util(num / 10, revNum);
        }
         // Check if first digit of num and revNum are equal
        if (num % 10 == revNum % 10) {
             // if yes, revNum will move with num
             return revNum / 10;
        }
else {
             // exit
             throw new Exception();
        }
```

```
    }
    //method to check if a given number is palindrome using palindrome utility function
    public static int isPalindrome(int num)  throws Exception {
        if (num < 0)
            num = (-num);

        int revNum = (num);
        return isPalindrome_util(num, revNum);
    }

    public static void main(String args[]) {
        int n = 1242;
        try {
            isPalindrome(n);
            System.out.println("Yes, the given number " + n + " is a palindrome.");
        } catch (Exception e) {
            System.out.println("No, the given number " + n + " is not a palindrome.");
        }
        n = 1221;
        try {
            isPalindrome(n);
            System.out.println("Yes, the given number " + n + " is a palindrome.");
        } catch (Exception e) {
            System.out.println("No, the given number " + n + " is not a palindrome.");
        }
    }
}
```

Keluaran

```
No, the given number 1242 is not a palindrome.

Yes, the given number 1221 is a palindrome.
```

3) Reverse String Recursion Java

Diberikan string "Hello" kita harus membalikkannya sehingga string yang dihasilkan adalah "olleH".

Ini dilakukan dengan menggunakan Rekursive. Mulai dari karakter terakhir dalam string kita secara rekursif mencetak setiap karakter hingga semua karakter dalam string habis.

Program di bawah ini menggunakan Rekursive untuk membalikkan string yang diberikan.

```
class String_Reverse
{
    //recursive method to reverse a given string
    void reverseString(String str)
    {
        //base condition; return if string is null or with 1 or less character
        if ((str==null)||(str.length() <= 1))
           System.out.println(str);
        else
        {
            //recursively print each character in the string from the end
            System.out.print(str.charAt(str.length()-1));
            reverseString(str.substring(0,str.length()-1));
        }
    }
}
class Main{
    public static void main(String[] args)
    {
        String inputstr = "SoftwareTestingHelp";
        System.out.println("The given string: " + inputstr);
        String_Reverse obj = new String_Reverse();
        System.out.print("The reversed string: ");
        obj.reverseString(inputstr);

    }
}
```

Keluaran

```
The given string: SoftwareTestingHelp

The reversed string: pleHgnitseTerawtfoS
```

4) Pencarian Biner Rekursive Java

Algoritme pencarian biner adalah algoritma yang terkenal untuk pencarian. Dalam algoritme ini, jika diberi larik yang diurutkan berisi n elemen, kita mencari larik ini untuk elemen kunci yang diberikan. Pada awalnya, kita membagi array menjadi dua bagian dengan mencari elemen tengah dari array tersebut.

Kemudian tergantung pada apakah kunci <mid atau key> mid kita membatasi pencarian kita di paruh pertama atau kedua dari larik. Dengan cara ini, proses yang sama diulangi hingga lokasi elemen kunci ditemukan.

Kita akan menerapkan algoritma ini menggunakan Rekursive di sini.

```
import java.util.*;
class Binary_Search {
    // recursive binary search
    int binarySearch(int numArray[], int left, int right, int key)   {
        if (right >= left) {
            //calculate mid of the array
            int mid = left + (right - left) / 2;
             // if the key is at mid, return mid
            if (numArray[mid] == key)
                return mid;
             // if key < mid, recursively search the left subarray
            if (numArray[mid] > key)
                return binarySearch(numArray, left, mid - 1, key);
             // Else recursively search in the right subarray
            return binarySearch(numArray, mid + 1, right, key);
        }
        // no elements in the array, return -1
        return -1;
    }
}
class Main{
    public static void main(String args[])   {
        Binary_Search ob = new Binary_Search();
        //declare and print the array
        int numArray[] = { 4,6,12,16,22};
        System.out.println("The given array : " + Arrays.toString(numArray));
        int len = numArray.length;              //length of the array
        int key = 16;                           //key to be searched
        int result = ob.binarySearch(numArray, 0, len - 1, key);
        if (result == -1)
            System.out.println("Element " + key + " not present");
        else
            System.out.println("Element " + key + " found at index " + result);
    }
}
```

Keluaran

```
The given array : [4, 6, 12, 16, 22]

Element 16 found at index 3
```

5) Temukan Nilai Minimum Dalam Array Menggunakan Rekursive

Menggunakan Rekursive kita juga dapat menemukan nilai minimum dalam array.

Program Java untuk menemukan nilai minimum dalam larik diberikan di bawah ini.

```
import java.util.*;
class Main {
     static int getMin(int numArray[], int i, int n)
     {
         //return first element if only one element or minimum of the array elements
         return (n == 1) ? numArray[i] :
             Math.min(numArray[i], getMin(numArray,i + 1 , n - 1));
     }
public static void main(String[] args) {
          int numArray[] = { 7,32,64,2,10,23 };
          System.out.println("Given Array : " + Arrays.toString(numArray));
          int n = numArray.length;
          System.out.print("Minimum element of array: " + getMin(numArray, 0, n) + "\n");
      }
}
```

Keluaran

```
Given Array : [7, 32, 64, 2, 10, 23]

Minimum element of array: 2
```

Ini adalah beberapa contoh Rekursive. Terlepas dari contoh-contoh ini, banyak masalah lain dalam perangkat lunak yang dapat diimplementasikan menggunakan teknik rekursif.

Jenis Rekursive

Rekursive terdiri dari dua jenis berdasarkan kapan panggilan dilakukan ke metode rekursif.

Yaitu:

1) Rekursive Ekor

Ketika panggilan ke metode rekursif adalah pernyataan terakhir yang dieksekusi di dalam metode rekursif, ini disebut "Rekursive Ekor".

Dalam Rekursive ekor, pernyataan panggilan rekursif biasanya dijalankan bersama dengan pernyataan kembalian dari metode tersebut.

Sintaks umum untuk Rekursive ekor diberikan di bawah ini:

```
methodName ( T parameters…){
{
    if (base_condition == true)
    {
        return result;
    }
    return methodName (T parameters …)      //tail recursion
}
```

2) Rekursive Kepala

Rekursive kepala adalah setiap pendekatan rekursif yang bukan merupakan Rekursive ekor. Jadi, bahkan Rekursive umum adalah Rekursive di depan.

Sintaks Rekursive head adalah sebagai berikut:

```
methodName (T parameters…){
    if (some_condition == true)
    {
        return methodName (T parameters…);
    }
    return result;
}
```

Rekursive Vs Iterasi Di Jawa

Pengulangan	Pengulangan
Rekursive adalah proses di mana metode memanggil dirinya sendiri berulang kali hingga kondisi dasar terpenuhi.	Iterasi adalah proses di mana sepotong kode dieksekusi berulang kali untuk beberapa kali tertentu atau hingga suatu kondisi terpenuhi.
Apakah aplikasi untuk fungsi.	Dapat diterapkan untuk loop.
Berfungsi dengan baik untuk ukuran kode yang lebih kecil.	Berfungsi dengan baik untuk ukuran kode yang lebih besar.
Memanfaatkan lebih banyak memori karena setiap panggilan rekursif didorong ke stack	Memori yang digunakan relatif lebih sedikit.
Sulit untuk men-debug dan memelihara	Lebih mudah untuk men-debug dan memelihara
Menghasilkan luapan tumpukan jika kondisi dasar tidak ditentukan atau tidak tercapai.	Dapat dieksekusi tanpa batas tetapi pada akhirnya akan menghentikan eksekusi dengan kesalahan memori apa pun
Kompleksitas waktu sangat tinggi.	Kompleksitas waktu relatif lebih rendah.

Pertanyaan yang Sering Diajukan

Q # 1) Bagaimana cara kerja Rekursive di Java?

Jawaban: Dalam Rekursive, fungsi rekursif memanggil dirinya sendiri berulang kali hingga kondisi dasar terpenuhi. Memori untuk fungsi yang dipanggil didorong ke tumpukan di bagian atas memori untuk fungsi panggilan. Untuk setiap pemanggilan fungsi, salinan terpisah dari variabel lokal dibuat.

Q # 2) Mengapa Rekursive digunakan?

Jawaban: Rekursive digunakan untuk memecahkan masalah yang dapat dipecah menjadi masalah yang lebih kecil dan keseluruhan masalah dapat dinyatakan dalam masalah yang lebih kecil.

Rekursive juga digunakan untuk masalah-masalah yang terlalu kompleks untuk diselesaikan dengan menggunakan pendekatan berulang. Selain masalah yang bukan merupakan masalah kompleksitas waktu, gunakan Rekursive.

Q # 3) Apa manfaat Rekursive?

Menjawab:

Manfaat Rekursive meliputi:

1. Rekursive mengurangi pemanggilan fungsi yang berlebihan.
2. Rekursive memungkinkan kita menyelesaikan masalah dengan mudah jika dibandingkan dengan pendekatan berulang.

Q # 4) Mana yang lebih baik - Rekursive atau Iterasi?

Jawaban: Rekursive membuat panggilan berulang sampai fungsi dasar tercapai. Jadi ada overhead memori sebagai memori untuk setiap panggilan fungsi didorong ke stack.

Iterasi di sisi lain tidak memiliki banyak overhead memori. Eksekusi Rekursive lebih lambat daripada pendekatan iteratif. Rekursive mengurangi ukuran kode sementara pendekatan berulang membuat kode menjadi besar.

Q # 5) Apa Keuntungan dari Rekursive dibandingkan Iterasi?

Menjawab:

- Rekursive membuat kode lebih jelas dan lebih pendek.
- Rekursive lebih baik daripada pendekatan berulang untuk masalah seperti Menara Hanoi, penjelajahan pohon, dll.
- Karena setiap panggilan fungsi memiliki memori yang didorong ke tumpukan, Rekursive menggunakan lebih banyak memori.

- Performa Rekursive lebih lambat daripada pendekatan iteratif.

Kesimpulan

Rekursive adalah konsep yang sangat penting dalam perangkat lunak terlepas dari bahasa pemrogramannya. Rekursive sebagian besar digunakan dalam memecahkan masalah struktur data seperti menara Hanoi, penjelajahan pohon, daftar tertaut, dll. Meskipun membutuhkan lebih banyak memori, Rekursive membuat kode lebih sederhana dan lebih jelas.

Kita telah menjelajahi semua tentang Rekursive dalam Kelas ini. Kita juga telah menerapkan banyak contoh pemrograman untuk pemahaman konsep yang lebih baik.

Latihan

SEGI TIGAS PASCAL

```
import java.util.Scanner;
public class RecursionPascalTriangle
{
   public static void display(int num)
   {
      for(int a = 0; a < num; a++)
      {
         for(int b = 0; b <= a; b++)
         {
            System.out.println(pascalTriangle(a, b) + " ");
         }
         System.out.println();
      }
   }
   public static int pascalTriangle(int a, int b)
   {
      if(b == 0 || b == a)
      {
         return 1;
      }
      else
      {
         return pascalTriangle(a - 1, b - 1) + pascalTriangle(a - 1, b);
      }
   }
   public static void main(String[] args)
   {
      Scanner sc = new Scanner(System.in);
      System.out.println("Please enter number of rows: ");
      int row = sc.nextInt();
      display(row);
      sc.close();
   }
}
```

Output:

Please enter number of rows: 8

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1

REKURSIVE
ALGORITMA dan PEMROGRAMAN (ADP)

Edited by: Gogor C. Setyawan