[2] Rekursi dan Looping

Fungsi Rekursif

✧ Fungsi rekursif dalam pemrograman merupakan fungsi yang memanggil/mendefinisikan dirinya sendiri selama kondisi pemanggilan dipenuhi.

-- contoh kasus: perkalian 2 buah angka

Perkalian	Penjumlahan Berulang	Hasil
1 x 4 =	4	4
2 x 4 =	4+4	8
3 x 4 =	4+4+4	12
4 x 4 =	4+4+4+4	16
5 x 4 =	4+4+4+4+4	20
6 x 4 =	4+4+4+4+4+4	24
7 x 4 =	4+4+4+4+4+4+4	28
8 x 4 =	4+4+4+4+4+4+4+4	32
9 x 4 =	4+4+4+4+4+4+4+4+4	36
10 x 4 =	4+4+4+4+4+4+4+4+4+4	40

Fungsi rekursif kali() untuk menghitung hasil kali dari dua bilangan:

```
public class FungsiRekursif {
    public static void main(String[] args) {
        //contoh perkalian
        System.out.println("2 x 4 = " + kali(2,4));
    }

    /*
        fungsi rekursif yang menghitung hasil perkalian 2 buah angka
        contoh: angka a dan b
        diambil dari konsep matematka: perkalian merupakan penjumlahan berulang
    */
    static int kali(int a, int b){
        //if a dikalikan dengan angka 1
        if(b==1)
            return a;
        //else
        return a + kali(a, b - 1);
    }
```

```
}
```

Mari kita uraikan langkah pemanggilannya:

```
kali(2, 4)
  -> 2 + kali(2, 3)
  -> 2 + (2 + kali(2, 2))
  -> 2 + (2 + (2 + kali(2, 1)))
  -> 2 + (2 + (2 + 2))
  -> 2 + (2 + 4)
  -> 2 + 6
  -> 8
```

Komponen Fungsi Rekursif

1. Kondisi kapan berhentinya fungsi

   ialah kondisi yang menyatakan kapan fungsi tersebut berhenti. Kondisi ini harus dapat dibuktikan akan tercapai, jika tidak maka fungsi tidak akan berhenti karena ada salah algoritma
2. Pengurangan atau pembagian data ketika fungsi memanggil dirinya sendiri

   fungsi rekursif selalu memanggil dirinya sendiri sambil mengurangi atau memecahkan data masukan setiap panggilannya. Hal ini penting diingat, karena tujuan utama dari rekursif ialah memecahkan masalah dengan mengurangi masalah tersebut menjadi masalah-masalah kecil.

Apa bedanya rekursif dan iterasi/perulangan?

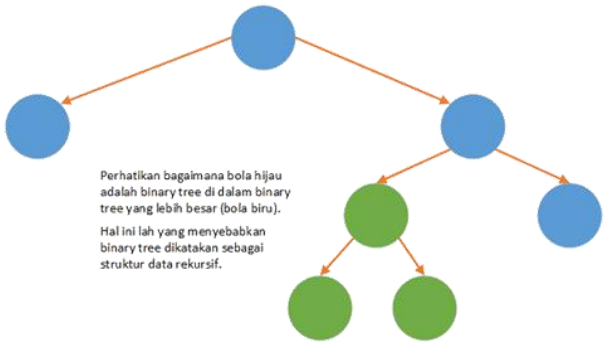
(catatan: looping/perulangan merupakan implementasi dari iterasi)

- ✧ Pendekatan rekursif memecah-mecah masalah untuk kemudian menyelesaikan masalah sedikit demi sedikit, dan inputan/parameternya pasti berbeda ketika memanggil dirinya sendiri lagi (lihat contoh program diatas)
- ✧ Pendekatan iteratif justru langsung mencoba menyelesaikan masalah, tanpa memecah-mecahkan masalah tersebut menjadi lebih kecil terlebih dahulu

Kapan kita harus menggunakan rekursif?

Berikut ini beberapa contoh penggunaan algoritma rekursif:

- ✧ menyelesaikan permasalahan di matematika,
  - menghitung phi dengan membagi keliling lingkaran dengan diameternya
  - faktorial
- ✧ penelurusan data di dalam sebuah binary tree.



sebuah binary tree, yang dapat didefinisikan sebagai sebuah pohon dengan jumlah cabang yang selalu dua, secara alami adalah struktur data rekursif.

Algoritma rekursif sangat tepat diterapkan untuk permasalahan yang alaminya memang rekursif, seperti list dan pohon (tree), aplikasi games, merge sort, dan quick sort. Tree akan kita pelajari dipraktikum berikut-berikutnya :D

Looping

foreach

for-each merupakan salah satu perulangan di Java yang umumnya digunakan untuk array atau class Collection di Java (ArrayList, dll).

sintaks for-each:	setara dengan:

for (tipe_data var : array) {
  //statements using var;
}
	for (**int** i=0; I < arr.length; i++) {
	  tipe_data var = arr[i];
	  //statements using var;
	}

Hal yang kd bisa kita lakukan dengan for-each:

For-each loops are not appropriate when you want to modify the array
 	
for (int num : marks)
{
// only changes num, not the array element
num = num*2;
}
	For-each loops do not keep track of index. So we can not obtain array index using For-Each loop
	
	for (int num : numbers)
	{
	if (num == target)
	{
	return ???; // do not know the index of num
	}
	}

For-each only iterates forward over the array in single steps
 	
// cannot be converted to a for-each loop
for (int i=numbers.length-1; i>0; i--)
{
System.out.println(numbers[i]);
}
	For-each cannot process two decision making statements at once
	
	// cannot be easily converted to a for-each loop
	for (int i=0; i<numbers.length; i++)
	{
	if (numbers[i] == arr[i])
	{ ...
	}
	}

-- contoh kasus: mencari angka dengan nilai terbesar (maks) dalam array

```
class BelajarForeach{
  public static void main(String[] arg){
    {
      int[] marks = { 125, 132, 95, 116, 110 };

      int highest_marks = maximum(marks);
      System.out.println("Nilai tertinggi adalah " + highest_marks);
    }
  }
  public static int maximum(int[] numbers){
    int maxSoFar = numbers[0];

    // for each loop
    for (int num : numbers)
    {
      if (num > maxSoFar) {
        maxSoFar = num;
      }
    }
    return maxSoFar;
  }
}
```