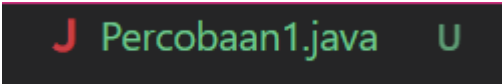


NAMA : FIFA NUURUN HALIZAH
NIM : 244107020019
NO. ABSEN : 09
KELAS : 1D

JOBSHEET 12

Percobaan 1: Membuat Fungsi Tanpa Parameter

1. Buat file Java dengan nama Percobaan1



2. Buat fungsi static dengan nama faktorialRekursif()

```
public class Percobaan1 {  
  
    static int faktorialRekursif(int n) {  
        if (n == 0) {  
            return 1;  
        } else {  
            return n * faktorialRekursif(n - 1);  
        }  
    }  
}
```

3. Buat lagi fungsi static dengan nama faktorialIteratif()

```
static int faktorialIteratif(int n) {  
    int faktor = 1;  
    for (int i = n; i >= 1; i--) {  
        faktor = faktor * i;  
    }  
    return faktor;  
}
```


4. Buatlah fungsi main dan lakukan pemanggilan terhadap kedua fungsi yang telah dibuat sebelumnya, dan tampilkan hasil yang didapatkan.

```
Run | Debug  
public static void main(String[] args) {  
    System.out.println(faktorialRekursif(n:5));  
    System.out.println(faktorialIteratif(n:5));  
}
```

5. Compile dan run program



6. Commit dan push kode program ke Github

 Percobaan1.java

percobaan1

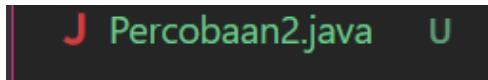
2 menit yang lalu

Pertanyaan!

1. Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri untuk menyelesaikan masalah dengan cara membagi masalah besar menjadi submasalah yang lebih kecil.
2. Kasus penggunaan fungsi rekursif dalam pemrograman yaitu menghitung faktorial, Traversing Pohon, Deret Fibonacci, Mencari Elemen dalam Array, Permutasi atau Kombinasi, dan lain sebagainya.
3. Ya, hasil yang diberikan oleh fungsi faktorialRekursif() dan faktorialIteratif() pada program Percobaan1 sama. Perbedaan alur program antara fungsi rekursif dan iteratif terletak pada cara perulangan dilakukan. Fungsi rekursif memanggil dirinya sendiri berulang kali, menyimpan status setiap pemanggilan di stack hingga mencapai base case, lalu menghitung hasil saat proses kembali (*unwinding*). Sedangkan, Fungsi iteratif menggunakan loop (for/while) untuk perulangan langsung tanpa stack, sehingga alurnya linear dan lebih efisien memori.

Percobaan 2

1. Buat file Java dengan nama Percobaan2



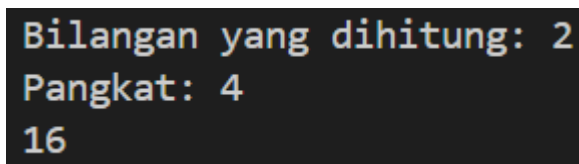
2. Buat fungsi static dengan nama hitungPangkat()

```
public class Percobaan2 {  
    static int hitungPangkat (int x, int y) {  
        if (y == 0) {  
            return (1);  
        } else {  
            return (x * hitungPangkat(x, y - 1));  
        }  
    }  
}
```

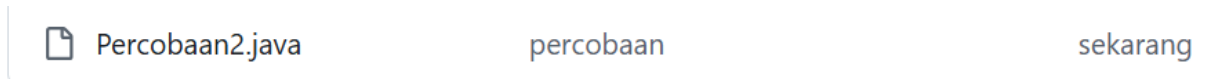
3. Tambahkan kode berikut ini untuk menerima input dari keyboard

```
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
    int bilangan, pangkat;  
  
    System.out.print(s:"Bilangan yang dihitung: ");  
    bilangan = sc.nextInt();  
    System.out.print(s:"Pangkat: ");  
    pangkat = sc.nextInt();  
    System.out.println(hitungPangkat(bilangan, pangkat));  
}
```

4. Compile dan run program



5. Commit dan push kode program ke Github



Pertanyaan !

1. Proses pemanggilan fungsi rekursif `hitungPangkat(bilangan, pangkat)` akan terus dilakukan hingga memenuhi kondisi base case. Kondisi base case terjadi ketika nilai `y` (pangkat) sama dengan 0, di mana fungsi akan mengembalikan nilai 1 tanpa melakukan pemanggilan rekursif lebih lanjut.
2. Tambahkan kode program untuk mencetak deret perhitungan pangkatnya.

```
import java.util.Scanner;

public class Percobaan2 {
    static int hitungPangkat(int x, int y) {
        if (y == 0) {
            System.out.print(s:"1");
            return 1;
        } else {
            System.out.print(x + "x");
            return x * hitungPangkat(x, y - 1);
        }
    }

    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int bilangan, pangkat;

        System.out.print(s:"Bilangan yang dihitung: ");
        bilangan = sc.nextInt();
        System.out.print(s:"Pangkat: ");
        pangkat = sc.nextInt();



        System.out.print(s:"Perhitungan: ");
        int hasil = hitungPangkat(bilangan, pangkat);
        System.out.println(" = " + hasil);
    }
}
```

Outputnya

```
Bilangan yang dihitung: 2
Pangkat: 5
Perhitungan: 2x2x2x2x2x1 = 32
```

Percobaan 3

1. Buat file java dengan nama percobaan3

 Percobaan3.java 

2. Masukkan kode program yang sudah ditentukan

```
import java.util.Scanner;

public class Percobaan3 {
    static double hitungLaba (double saldo, int tahun) {
        if (tahun == 0) {
            return (saldo);
        } else {
            return (1.11 * hitungLaba(saldo, tahun - 1));
        }
    }
}

Run | Debug
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    double saldoAwal;
    int tahun;


    System.out.print(s:"Jumlah saldo awal: ");
    saldoAwal = sc.nextInt();
    System.out.print(s:"Lamanya investasi (tahun): ");
    tahun = sc.nextInt();

    System.out.print("Jumlah saldo setelah " + tahun + " tahun : ");
    System.out.print(hitungLaba(saldoAwal, tahun));
}
}
```

3. Compile dan run program

```
Jumlah saldo awal: 500000
Lamanya investasi (tahun): 1
Jumlah saldo setelah 1 tahun : 555000.0
```

4. Commit dan push kode program ke Github

| | | |
|---|-----------|----------|
|  Percobaan3.java | percobaan | sekarang |
|---|-----------|----------|

Pertanyaan!

1. Pada program Percobaan3, base case terjadi ketika nilai `tahun` sama dengan 0, yang menyebabkan fungsi `hitungLaba()` mengembalikan saldo tanpa perhitungan lebih lanjut (`if (tahun == 0) { return saldo; }`). Sedangkan, recursion call terjadi ketika fungsi memanggil dirinya sendiri dengan parameter `tahun - 1` untuk menghitung saldo pada tahun sebelumnya, dan hasilnya dikalikan dengan 1.11 untuk memperoleh saldo setelah bunga pada setiap tahun (`return (1.11 * hitungLaba(saldo, tahun - 1));`).
2. Pada algoritma rekursif `hitungLaba(100000, 3)`, fase ekspansi adalah proses memecah masalah menjadi submasalah yang lebih kecil melalui pemanggilan fungsi secara berulang hingga mencapai **base case**. Berikut jejaknya:

1. Ekspansi:

- `hitungLaba(100000, 3)` → `1.11 * hitungLaba(100000, 2)`
- `hitungLaba(100000, 2)` → `1.11 * hitungLaba(100000, 1)`
- `hitungLaba(100000, 1)` → `1.11 * hitungLaba(100000, 0)`
- `hitungLaba(100000, 0)` = 100000 (**base case**)

2. Substitusi Proses kembali (unwinding) menghitung hasil dari submasalah:

- `hitungLaba(100000, 0)` = 100000
- `hitungLaba(100000, 1)` = `1.11 * 100000 = 111000`
- `hitungLaba(100000, 2)` = `1.11 * 111000 = 123210`
- `hitungLaba(100000, 3)` = `1.11 * 123210 = 136761`

Pada fase substitusi, hasil base case digunakan untuk menghitung laba dari tahun ke tahun hingga kembali ke tahun ke-3. Hasil akhirnya adalah 136761.

TUGAS !

1. Buatlah program untuk menampilkan bilangan n sampai 0 dengan menggunakan fungsi rekursif dan fungsi iteratif. (DeretDescendingRekursif).

```
public class Tugas1 {
    static void deretRekursif(int n) {
        if (n < 0) {
            return;
        }
        System.out.print(n + " ");
        deretRekursif(n - 1);
    }
    static void deretIteratif(int n) {
        for (int i = n; i >= 0; i--) {
            System.out.print(i + " ");
        }
    }
}

Run | Debug
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print(s:"Masukkan nilai n: ");
    int n = sc.nextInt();

    System.out.println(x:"\nDeret Descending menggunakan Rekursif:");
    deretRekursif(n);

    System.out.println(x:"\n\nDeret Descending menggunakan Iteratif:");
    deretIteratif(n);
}
```

Outputnya

```
Masukkan nilai n: 4

Deret Descending menggunakan Rekursif:
4 3 2 1 0

4 3 2 1 0

Deret Descending menggunakan Iteratif:
4 3 2 1 0
```

2. Buatlah program yang di dalamnya terdapat fungsi rekursif untuk menghitung penjumlahan bilangan. Misalnya f = 8, maka akan dihasilkan 1+2+3+4+5+6+7+8 = 36 (PenjumlahanRekursif).

```

public class PenjumlahanRekursif {
    static int hitungPenjumlahan(int n, StringBuilder deret) {
        if (n == 1) {
            return 0;
        } else {
            int result = n + hitungPenjumlahan(n - 1, deret);

            if (n > 1) {
                deret.insert(offset:0, n + "+");
            } else {
                deret.insert(offset:0, n);
            }
            return result;
        }
    }
}

Run | Debug
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int f;

    System.out.print(s:"Masukkan nilai f: ");
    f = sc.nextInt();

    StringBuilder deret = new StringBuilder();

    int hasil = hitungPenjumlahan(f, deret);
    System.out.println("Deret penjumlahan: " + deret + " = " + hasil);
}

```

Outputnya

```

Masukkan nilai f: 8
Deret penjumlahan: 8+7+6+5+4+3+2+1 = 36

```

3. Sepasang marmut yang baru lahir (jantan dan betina) ditempatkan pada suatu pembiakan. Setelah dua bulan pasangan marmut tersebut melahirkan sepasang marmut kembar (jantan dan betina). Setiap pasangan marmut yang lahir juga akan melahirkan sepasang marmut juga setiap 2 bulan. Berapa pasangan marmut yang ada pada akhir bulan ke-12? Buatlah programnya menggunakan fungsi rekursif! (Fibonacci). Berikut ini adalah ilustrasinya dalam bentuk tabel.

```

public class marmut09 {
    static int hitungPasangan(int bulan) {
        if (bulan == 1 || bulan == 2) {
            return 1;
        } else {
            return hitungPasangan(bulan - 1) + hitungPasangan(bulan - 2);
        }
    }
}

Run | Debug
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print(s:"Masukkan bulan ke-: ");
    int bulan = sc.nextInt();

    int totalPasangan = hitungPasangan(bulan);
    System.out.println("Jumlah total pasangan marmut pada bulan ke-" + bulan + " adalah: " + totalPasangan);
}

```

Outputnya

Masukkan bulan ke-: 6

Jumlah total pasangan marmut pada bulan ke-6 adalah: 8