# LAPORAN PRAKTIKUM

# JOBSHEET PERTEMUAN 14


NADYA AURORA GEBI AGISTA

NIM 244107020034

KELAS TI 1H

**PROGRAM STUDI D-IV TEKNIK INFORMATIKA**

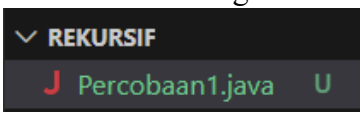
**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2024**

## PERCOBAAN 1

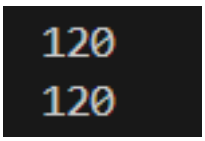
1. Buat file baru dengan nama Percobaan1




2. Buat fungsi static dengan nama faktorialRekursif()

```java
public class Percobaan1 {
    static int faktorialRekursif(int n) {
        if (n == 0) {
            return (1);
        } else {
            return (n * faktorialRekursif(n-1));
        }
    }
    static int faktorialIteratif(int n) {
        int faktor = 1;
        for (int i = n; i >=1; i--) {
            faktor = faktor * i;
        }
        return faktor;
    }
    public static void main(String[] args) {
        System.out.println(faktorialRekursif(5));
        System.out.println(faktorialIteratif(5));
    }
}
```

3. Hasil Output :




## JAWABAN PERTANYAAN

1. Fungsi rekursif adalah sebuah fungsi yang terdapat perintah untuk memanggil fungsi itu sendiri (dirinya sendiri). Dengan demikian, proses pemanggilan fungsi akan terjadi secara berulang-ulang.
2. Contoh kasus penggunaan fungsi rekursif, yaitu menghitung faktorial, menghitung Fibonacci, mencari bilangan prima, dan menghitung traversal.
3. Hasil yang diberikan oleh fungsi faktorialRekursif() dan faktorialIteratif() memiliki hasil yang sama. Keduanya menghasilkan nilai 120.

- **Perbedaan Alur Fungsi Rekursif dan Iteratif**
    1. Rekursif
        - Fungsi rekursif bekerja dengan memanggil dirinya sendiri secara berulang dengan mengurangi nilai n hingga mencapai base case.
        - Menggunakan stack untuk menyimpan setiap pemanggilan.
        - Setelah mencapai base case, fungsi akan mulai kembali (unwind) ke pemanggil sebelumnya.
    2. Iteratif
        - Fungsi Iteratif menggunakan perulangan for untuk menghitung faktorial.
        - Iterasi berjalan dari n hingga 1, mengalikan nilai faktor dengan indeks iterasi pada setiap langkah.
        - Setelah iterasi selesai, fungsi akan mengembalikan hasil akhir.
        - Tidak memerlukan stack tambahan

**PERCOBAAN 2**

1. Kode Program

```java
import java.util.Scanner;
public class Percobaan2 {
    static int hitungPangkat(int x, int y) {
        if (y == 0) {
            return (1);
        } else {
            return (x * hitungPangkat(x, y - 1));
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        int bilangan, pangkat;
        System.out.print("Bilangan yang dihitung: ");
        bilangan = sc.nextInt();
        System.out.print("Pangkat: ");
        pangkat = sc.nextInt();
        System.out.println(hitungPangkat(bilangan, pangkat));
    }
}
```

2. Hasil Output :

```
Bilangan yang dihitung: 5
Pangkat: 2
25
```

**JAWABAN PERTANYAAN**

1. Pemanggilan rekursif berhenti setelah fungsi hitungPangkat() dipanggil sebanyak $y+1$ kali, karena setiap langkah rekursi mengurangi y sebesar 1 hingga mencapai 0.
2. Modifikasi Kode Program :

```java
import java.util.Scanner;

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

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

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

Hasil Output :

```
Bilangan yang dihitung: 2
Pangkat: 5
Deret perhitungan: 2 * 2 * 2 * 2 * 2 * 1 = 32
```

**PERCOBAAN 3**

1. Kode Program :

```java
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));
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        System.out.print("Jumlah saldo awal : ");
        double saldoAwal = sc.nextDouble();
        System.out.print("Lamanya investasi (tahun) : ");
        int tahun = sc.nextInt();


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

2. Hasil Output :

```
Jumlah saldo awal : 10000
Lamanya investasi (tahun) : 1
Jumlah saldo setelah 1 tahun : 11100.000000000002
```

**JAWABAN PERTANYAAN**

1.

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

Base Case

Recursion Call

2. - **Trace fase 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)

- **Trace fase Substitusi :**

hitungLaba(100000, 1) → 1.11 * 100000 = 111000
hitungLaba(100000, 2) → 1.11 * 111000 = 123210
hitungLaba(100000, 3) → 1.11 * 123210 = 136965. 10000000003

**TUGAS**

1. Kode Program :

```java
import java.util.Scanner;
public class Tugas1{
    static void deretRekursif(int n) {
        if (n < 0) {
            return;
        } else {
            System.out.print(n + " ");
            deretRekursif(n - 1);
        }
    }
    static void deretIteratif(int n) {
        for (int i = n; i >= 0; i--) {
            System.out.print(i + " ");
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukkan nilai n: ");
        int n = sc.nextInt();
        System.out.println("Deret Rekursif:");
        deretRekursif(n);
        System.out.println("\nDeret Iteratif:");
        deretIteratif(n);
    }
}
```

Hasil Output :

```
Masukkan nilai n: 3
Deret Rekursif:
3 2 1 0
Deret Iteratif:
3 2 1 0
```

2. Kode Program :

```java
import java.util.Scanner;
public class Tugas2 {
    static int hitung(int n) {
        if (n == 1) {
            System.out.print("1");
            return 1;
        } else {
            int hasil = hitung(n-1);
            System.out.print( " + " + n);
            return hasil + n;
        }
    }

    public static void main(String[] args) {
        System.out.println("f = 8");
        System.out.println("Hasil penjumlahan : ");
        System.out.print(" = " + hitung(8));
    }
}
```

Hasil Output :

```
f = 8
Hasil penjumlahan :
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 = 36
```

3. Kode Program :

```java
import java.util.Scanner;
public class Tugas3 {
    static boolean cek(int n, int i) {
        if (i * i > n) {
            return true;
        }
        if (n % i == 0) {
            return false;
        }
        return cek(n, i + 1);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukkan bilangan untuk dicek apakah prima: ");
        int n = sc.nextInt();

        if (n >= 2 && cek(n, 2)) {
            System.out.println(n + " adalah bilangan prima.");
        } else {
            System.out.println(n + " bukan bilangan prima.");
        }
    }
}
```

Hasil Output :

```
Masukkan bilangan untuk dicek apakah prima: 4
4 bukan bilangan prima.
```

4. Kode Program :

```java
import java.util.Scanner;
public class Tugas4 {
    static int fibonacci(int n) {
        if (n <= 2) {
            return 1;
        }
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
    static void outputFibonacci(int n, int i) {
        if (i > n) {
            return;
        }
        System.out.print(fibonacci(i) + " ");
        outputFibonacci(n, i + 1);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukkan angka ke-n: ");
        int n = sc.nextInt();
        System.out.print("Deret Fibonacci ke-" + n + " : ");
        outputFibonacci(n, 1);
        System.out.println();
    }
}
```

Hasil Output :

```
Masukkan angka ke-n: 12
Deret Fibonacci ke-12 : 1 1 2 3 5 8 13 21 34 55 89 144
```