Nama : Irna Nurdiyani
NIM : 22060016
Prodi : Teknik Informatika / RP

Latihan
Penggunaan Deno Command line interface

```typescript
function greet(name: string): string {
    return `Hello, ${name}!`;
  }

console.log(greet("world"));

// 1
var a = 10;
var a = 20;
var b = 30;
console.log(a);
console.log(a);
console.log(b);

b = b + 10;

const c = "software";
console.log(c);

// 2
let isFinished: boolean = false;
console.log(isFinished, typeof isFinished); // <1>
let price: number = 150.34;
console.log(price, typeof price);
let numOfEmployees: number = 25;
console.log(numOfEmployees, typeof numOfEmployees);
let progLang: string = 'TypeScript';
console.log(progLang, typeof progLang)
let university: string[] = ['UT', 'UGM', 'ITB'];
console.log(university, typeof university)
let employee: [number, string, boolean, number, string];
employee = [1, 'Zaky Aditya', true, 20, 'Engineer'];
console.log(employee, typeof employee);
enum Color {
    Black = 2,
    Blue,
    Yellow,
    Green = 3,
    Red = 3 * 3
}
console.log(Color, typeof Color);

// 3
let code: string | number;
console.log(code, typeof code)

code = 'my code';
console.log(code, typeof code);
```

```ts
class Person { // <1>
    perNik: string; // <2>
    protected perNama: string;
    perAlamat!: string;
    constructor(nik: string, nama: string) { // <3>
        this.perNik = nik;
        this.perNama = nama;
    }

}
class Pegawai extends Person { // <4>
    pegNpp: string; // <5>
    private _pegJmlTanggungan!: number;
    readonly dept: string;
    gaji!: number;
    static potongPajak = 10; // <6>

    constructor(nik: string, npp: string, // <7>

        nama: string, dept: string) {
        super(nik, nama);
        this.pegNpp = npp;
        this.dept = dept;
    }
    getGaji(): number { // <8>
        return this.gaji;
    }
    setGaji(gajiBaru: number): void { // <9>
        this.gaji = gajiBaru;
    }
    getPotonganPajak(): number { // <10>
        return this.gaji * (Pegawai.potongPajak / 100);
    }
    presensi(): void { // <11>
        let dateTime = new Date();
        console.log("Presensi pada " +
            dateTime.toLocaleTimeString() +
        ' - ' + dateTime.toDateString());
    }
}

let pakBambang = new Pegawai('nik1122', 'npp123', // <12>
'Bambang Purnomosidi', 'IT');
console.log(pakBambang.setGaji(15750500)); // <13>
console.log(pakBambang.getGaji());
console.log(pakBambang.presensi());
console.log(pakBambang.getPotonganPajak());
```

```ts
interface IPerson { // <1>
    nik: string;
    nama: string;
    alamat: string;
    menikah: boolean;
}

interface IPegawai extends IPerson { // <2>
    readonly npp: string;
    jabatan: string;
    gaji: number;
    email?: string;
}

let peg01: IPegawai = { // <3>
    nik: '012345',
    nama: 'Donal',
    alamat: 'Jl. Awan Biru 21',
    menikah: true,
    npp: '98123',
    jabatan: 'Manager SDM',
    gaji: 15000000
}

console.log(peg01.nama, peg01.jabatan); // <4>
// error: Cannot assign to 'npp' because it is a read-only property
// peg01.npp = '981234';
interface IKamusList { // <5>
[index:string]:string
}
let strKamus: IKamusList = {}; // <6>
    strKamus['university'] = 'universitas';
    strKamus['freedom'] = 'merdeka';
    console.log(strKamus['university']);
    interface IPemrosesNilai { // <7>
    (kunci: number,
    nilai: string ): void
}

function tambahNilai (kunci: number, nilai: string): void { // <8>
    console.log('Menambah ', kunci, nilai);
}
function perbaruiNilai (kunci: number, nilaiBaru: string): void { // <9>
    console.log('Memperbarui ', kunci, nilaiBaru);
}
let pemrosesTambah: IPemrosesNilai = tambahNilai; // <10>
pemrosesTambah(123, 'Nilai 123');
let pemrosesPerbarui: IPemrosesNilai = perbaruiNilai;
pemrosesPerbarui(123, 'Nilai baru 123');
```