

Lembar Kerja Peserta Didik

Modul 4 - OOP Lanjutan

A. PETUNJUK Pengerjaan

1. LKPD ini dikerjakan secara berkelompok.
2. Kelompok menggunakan susunan anggota kelompok yang sudah ditetapkan di awal.
3. Sebelum kalian mengerjakan LKPD ini secara berkelompok, pastikan anda sudah bergabung di GitHub
4. LKPD diisi sesuai dengan perintah yang telah dijelaskan pada setiap soalnya.
5. Jika kalian belum memahami instruksi yang diberikan di dalam LKPD, mintalah penjelasan dari bapak/ibu guru.
6. Setelah selesai mengerjakan soal, persiapkan diri kalian untuk melakukan presentasi penjelasan kode.
7. **DIPERBOLEHKAN MENGGUNAKAN CHATGPT SELAMA PROSES Pengerjaan.**

B. POIN TAMBAHAN

1. Apabila kalian mengerjakan tutorial *visibility modifiers* yang diberikan pada **MODUL 4**.
2. Apabila kalian mengerjakan dan mengumpulkan tepat waktu (**WAKTU Pengerjaan 30 MENIT**).

C. PETUNJUK Pengumpulan

1. Pengumpulan tugas dilakukan perwakilan oleh ketua kelompok.
2. File yang dikumpulkan cukup dokumen ini saja dengan format .pdf.
 - a. Cara mendownload dokumen dengan format pdf:
 - i. Buka opsi file yang ada di pojok kiri atas layar.
 - ii. Kemudian pilih opsi download.
 - iii. Selanjutnya pilih tipe file PDF Documents.
3. Lalu kumpulkan file pada GitHub yang telah disediakan.

D. REFERENSI TAMBAHAN UNTUK BELAJAR

1. Kalian bisa melihat video dari YouTube berikut apabila kalian masih belum paham sepenuhnya mengenai materi yang dipelajari sebelumnya:
 - 1) Penjelasan *visibility modifiers* → [📺 Belajar Kotlin OOP - 27 Visibility Modifier](#)
 - 2) Tutorial dasar *visibility modifiers* → [📺 Tutorial Kotlin Dasar - 18. Visibility Modifiers](#)

D. SOAL

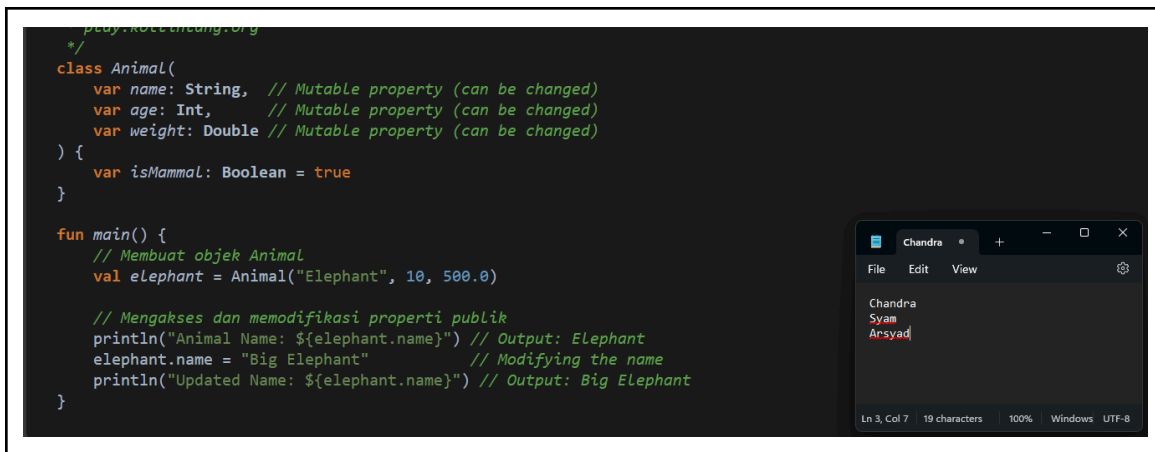
Nama Kelompok : Air Galon Mineral

Nama/No.Absen Anggota Kelompok :

- 1) Chandra
- 2) Arsyad
- 3) Syam

TUTORIAL 1 - TUTORIAL PENGGUNAAN VISIBILITY MODIFIERS PUBLIC PADA KOTLIN

1. Lampirkan gambar *screenshot* kode kalian dan sertakan notepad yang berisi nama anggota kelompok kalian!



The screenshot shows a Kotlin code editor with the following code:

```
play.kotlinlang.org
/**
 *
 */
class Animal(
    var name: String, // Mutable property (can be changed)
    var age: Int,      // Mutable property (can be changed)
    var weight: Double // Mutable property (can be changed)
) {
    var isMammal: Boolean = true
}

fun main() {
    // Membuat objek Animal
    val elephant = Animal("Elephant", 10, 500.0)

    // Mengakses dan memodifikasi properti publik
    println("Animal Name: ${elephant.name}") // Output: Elephant
    elephant.name = "Big Elephant"           // Modifying the name
    println("Updated Name: ${elephant.name}") // Output: Big Elephant
}
```

Overlaid on the bottom right is a Notepad window titled "Chandra" containing the names of the group members:

```
File Edit View
Chandra
Syam
Arsyad
```

The status bar at the bottom of the Notepad window shows "Ln 3, Col 7 19 characters 100% Windows UTF-8".

2. Lampirkan kode yang kalian buat.

```
class Animal(
    var name: String, // Mutable property (can be changed)
    var age: Int,      // Mutable property (can be changed)
    var weight: Double // Mutable property (can be changed)
) {
    var isMammal: Boolean = true
}

fun main() {
    // Membuat objek Animal
    val elephant = Animal("Elephant", 10, 500.0)

    // Mengakses dan memodifikasi properti publik
    println("Animal Name: ${elephant.name}") // Output:
Elephant
    elephant.name = "Big Elephant"           // Modifying the
name
```

```
println("Updated Name: ${elephant.name}") // Output: Big
Elephant
}
```

3. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

Kode yang diberikan mendefinisikan kelas **Animal** di Kotlin dengan beberapa properti dan metode dasar. Berikut adalah penjelasan cara kerjanya:

1. **Definisi Kelas:** Kelas **Animal** memiliki tiga properti mutable (**name**, **age**, dan **weight**) yang dapat diubah setelah objek dibuat. Juga terdapat properti **isMammal** yang diinisialisasi dengan nilai **true**.
2. **Membuat Objek:** Di dalam fungsi **main**, objek **elephant** dari kelas **Animal** dibuat dengan nama "Elephant", usia 10, dan berat 500.0.
3. **Mengakses Properti:** Program mencetak nama hewan menggunakan **println**, yang mengeluarkan "Animal Name: Elephant".
4. **Modifikasi Properti:** Nama hewan diubah menjadi "Big Elephant" menggunakan assignment, lalu dicetak kembali, yang menghasilkan "Updated Name: Big Elephant".

TUTORIAL 2 - TUTORIAL PENGGUNAAN VISIBILITY MODIFIERS PRIVATE PADA KOTLIN

1. Lampirkan gambar screenshot kode kalian dan sertakan notepad yang berisi nama anggota kelompok kalian!



```
class Animal(private var name: String, private var age: Int) {

    // Getter untuk name
    fun getName(): String {
        return name
    }

    // Setter untuk name
    fun setName(newName: String) {
        name = newName
    }

    // Getter untuk age
    fun getAge(): Int {
        return age
    }

    // Setter untuk age
    fun setAge(newAge: Int) {
        age = newAge
    }
}
```

The screenshot also shows a Notepad window titled 'Chandra' containing the names: Chandra, Syam, and Arsyad.

2. Lampirkan kode yang kalian buat.

```
class Animal(private var name: String, private var age: Int) {

    // Getter untuk name
    fun getName(): String {
```

```

        return name
    }

    // Setter untuk name
    fun setName(newName: String) {
        name = newName
    }

    // Getter untuk age
    fun getAge(): Int {
        return age
    }

    // Setter untuk age
    fun setAge(newAge: Int) {
        age = newAge
    }
}

fun main() {
    val myPet = Animal("Kucing", 2)

    // Menggunakan getter untuk mendapatkan nilai 'name'
    println("Nama hewan: ${myPet.getName()}") // Output: Kucing

    // Menggunakan setter untuk mengubah nilai 'name'
    myPet.setName("Banteng")
    println("Nama hewan setelah diubah: ${myPet.getName()}") // Output: Banteng

    // Menggunakan getter untuk mendapatkan nilai 'age'
    println("Umur hewan: ${myPet.getAge()}") // Output: 2

    // Menggunakan setter untuk mengubah nilai 'age'
    myPet.setAge(3)
    println("Umur hewan setelah diubah: ${myPet.getAge()}") // Output: 3
}

```

3. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

Getter dan Setter:

- **Getter untuk name:** Metode `getName()` mengembalikan nilai dari properti `name`.
- **Setter untuk name:** Metode `setName(newName: String)` memungkinkan pengubahan nilai `name` dengan parameter `newName`.

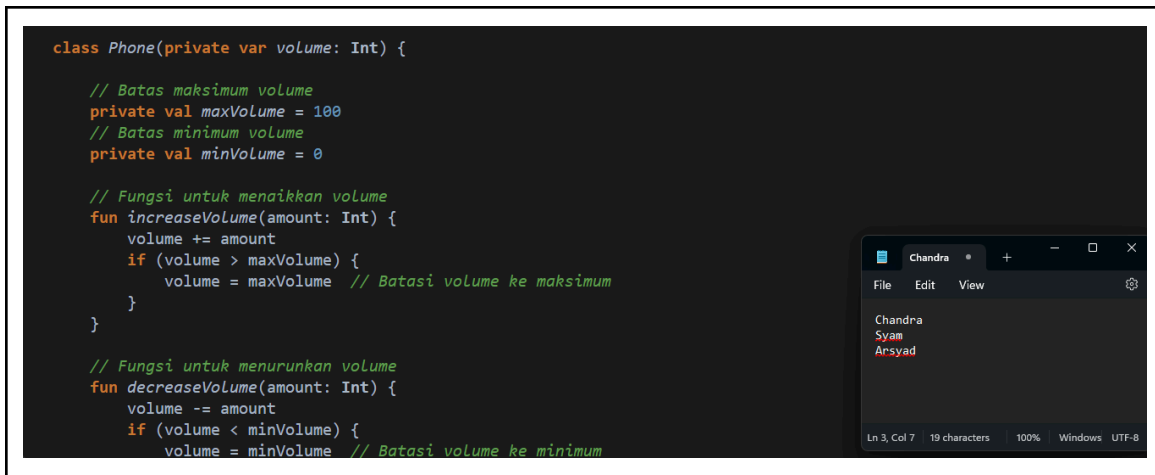
- **Getter untuk `age`:** Metode `getAge()` mengembalikan nilai dari properti `age`.
- **Setter untuk `age`:** Metode `setAge(newAge: Int)` memungkinkan pengubahan nilai `age` dengan parameter `newAge`.

Fungsi `main`:

- Sebuah objek `myPet` dari kelas `Animal` dibuat dengan nama "Kucing" dan umur 2.
- Menggunakan getter, program mencetak nama hewan ("Nama hewan: Kucing").
- Menggunakan setter, nama hewan diubah menjadi "Banteng", dan nama baru dicetak.
- Menggunakan getter, umur hewan dicetak ("Umur hewan: 2").
- Menggunakan setter, umur diubah menjadi 3, dan umur baru dicetak.

Studi kasus berapa kamu? - Nama studi kasusnya apa?

1. Lampirkan gambar screenshot kode kalian dan sertakan notepad yang berisi nama anggota kelompok kalian!



```
class Phone(private var volume: Int) {

    // Batas maksimum volume
    private val maxVolume = 100
    // Batas minimum volume
    private val minVolume = 0

    // Fungsi untuk menaikkan volume
    fun increaseVolume(amount: Int) {
        volume += amount
        if (volume > maxVolume) {
            volume = maxVolume // Batasi volume ke maksimum
        }
    }

    // Fungsi untuk menurunkan volume
    fun decreaseVolume(amount: Int) {
        volume -= amount
        if (volume < minVolume) {
            volume = minVolume // Batasi volume ke minimum
        }
    }
}
```

Chandra
Syam
Arsyad

2. Lampirkan kode yang kalian buat.

```
class Phone(private var volume: Int) {

    // Batas maksimum volume
    private val maxVolume = 100
    // Batas minimum volume
    private val minVolume = 0

    // Fungsi untuk menaikkan volume
    fun increaseVolume(amount: Int) {
        volume += amount
    }
}
```

```

        if (volume > maxVolume) {
            volume = maxVolume // Batasi volume ke maksimum
        }
    }

    // Fungsi untuk menurunkan volume
    fun decreaseVolume(amount: Int) {
        volume -= amount
        if (volume < minVolume) {
            volume = minVolume // Batasi volume ke minimum
        }
    }

    // Fungsi untuk mendapatkan volume saat ini
    fun getVolume(): Int {
        return volume
    }
}

fun main() {
    val myPhone = Phone(50) // Volume awal 50

    // Tampilkan volume awal
    println("Volume awal: ${myPhone.getVolume()}") // Output: 50

    // Naikkan volume sebesar 30
    myPhone.increaseVolume(30)

    // Tampilkan volume setelah naik
    println("Volume setelah naik: ${myPhone.getVolume()}") // Output: 80

    // Turunkan volume sebesar 70
    myPhone.decreaseVolume(70)

    // Tampilkan volume setelah turun
    println("Volume setelah turun: ${myPhone.getVolume()}") // Output: 10

    // Coba naikkan volume melewati batas maksimum
    myPhone.increaseVolume(100)

    // Tampilkan volume akhir
    println("Volume akhir: ${myPhone.getVolume()}") // Output: 100
}

```

3. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

Kelas **Phone**:

- Memiliki properti privat **volume** yang diinisialisasi dengan nilai awal yang diberikan saat objek dibuat.
- Terdapat dua batas: **maxVolume** (100) dan **minVolume** (0).
- **Metode `increaseVolume(amount: Int)`:**
 - Menambahkan **amount** ke **volume**.
 - Jika **volume** melebihi **maxVolume**, ia disetel ke **maxVolume**.
- **Metode `decreaseVolume(amount: Int)`:**
 - Mengurangi **amount** dari **volume**.
 - Jika **volume** kurang dari **minVolume**, ia disetel ke **minVolume**.
- **Metode `getVolume()`:**
 - Mengembalikan nilai **volume** saat ini.

Fungsi **main**:

- Membuat objek **myPhone** dengan volume awal 50.
- Mencetak volume awal.
- Menaikkan volume sebesar 30 dan mencetak volume baru.
- Menurunkan volume sebesar 70 dan mencetak volume baru.
- Mencoba menaikkan volume hingga 100 (melebihi batas maksimum) dan mencetak volume akhir.