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

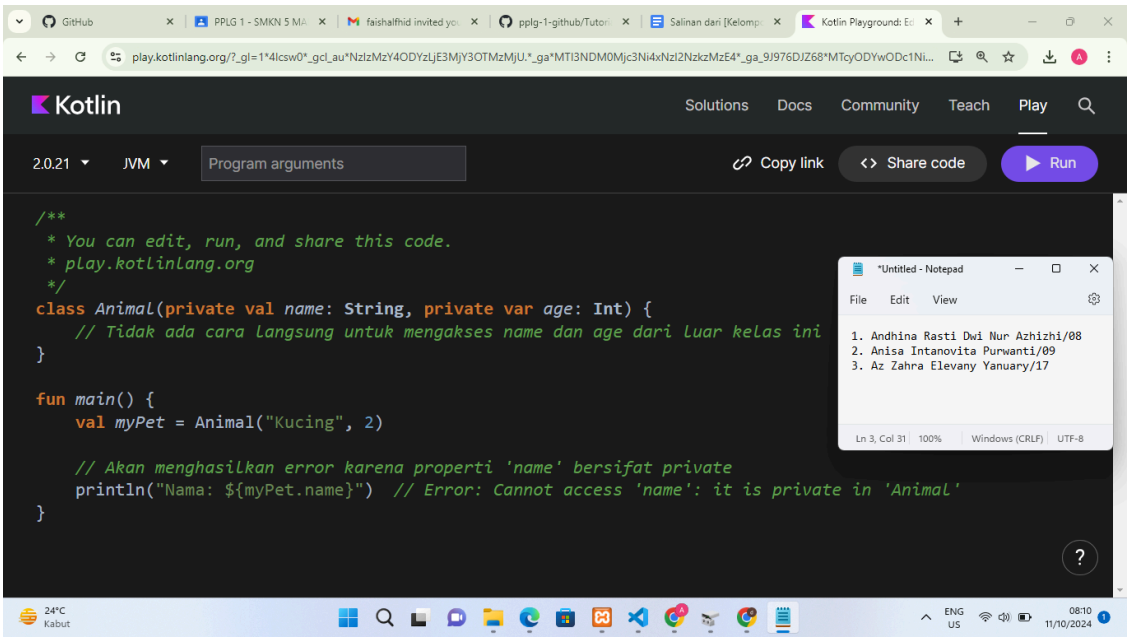
Nama Kelompok : .....

Nama/No.Absen Anggota Kelompok :

1) ...
2) ...
3) ...
4) ...
5) dst..

TUTORIAL 1 - TUTORIAL PENGGUNAAN VISIBILITY MODIFIERS PUBLIC PADA KOTLIN

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



Notes: Gambar hanya contoh, jadi bisa kalian hapus ya.

2. Lampirkan kode yang kalian buat.

```
   class Animal(private val name: String, private var age:
Int) {
    // Tidak ada cara langsung untuk mengakses name dan age
dari luar kelas ini
}

fun main() {
```

```
    val myPet = Animal("Kucing", 2)

    // Akan menghasilkan error karena properti 'name' bersifat
private
    println("Nama: ${myPet.name}")  // Error: Cannot access
'name': it is private in 'Animal'
}
```

Notes: Kode hanya contoh, jadi bisa kalian hapus ya.

3. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

Deklarasi Kelas `Animal`:

- Kelas `Animal` memiliki dua properti privat: `name` (tipe `String`) dan `age` (tipe `Int`).
- Karena kedua properti ini dideklarasikan sebagai `private`, mereka tidak dapat diakses langsung dari luar kelas.

Getter dan Setter:

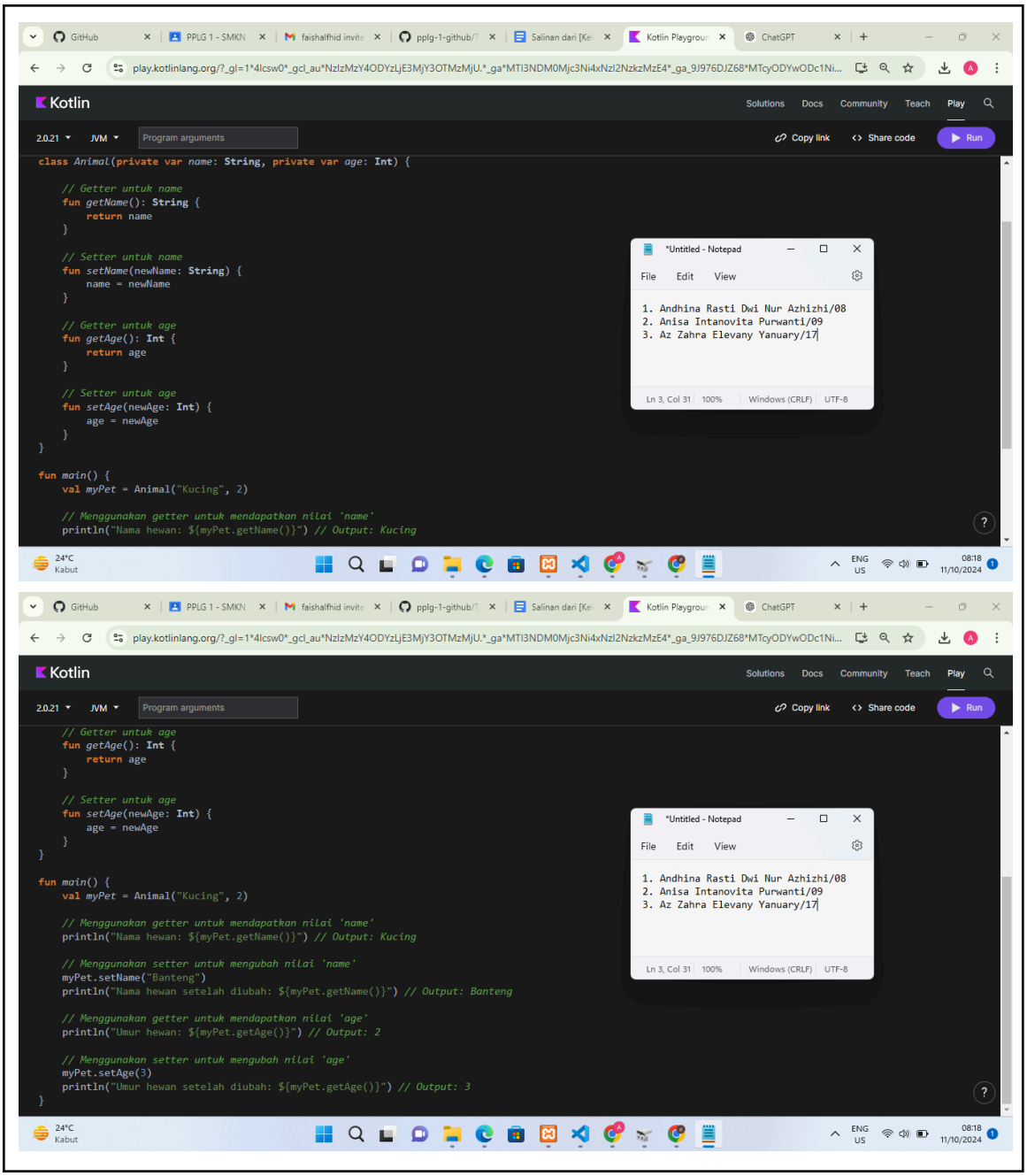
- Fungsi `getName()` dibuat untuk mengambil nilai dari properti `name`. Karena `name` bersifat `private`, fungsi ini memungkinkan pengguna kelas untuk membaca nilainya tanpa mengaksesnya langsung.
- Fungsi `getAge()` mengambil nilai dari properti `age`.
- Fungsi `setAge(newAge: Int)` memungkinkan pengguna untuk mengubah nilai properti `age` dari luar kelas.

`main` Function:

- Sebuah objek `myPet` dari kelas `Animal` dibuat dengan `name` berisi `"Kucing"` dan `age` berisi `2`.
- Kemudian, nama hewan peliharaan (`myPet.getName()`) dan umur awalnya (`myPet.getAge()`) ditampilkan menggunakan fungsi `println`.
- Nilai umur diubah menjadi `3` dengan memanggil `myPet.setAge(3)`.
- Umur baru ditampilkan menggunakan `myPet.getAge()`.

TUTORIAL 2 - TUTORIAL PENGGUNAAN VISIBILITY MODIFIERS PRIVATE PADA KOTLIN

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



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!

Kelas `Animal`:

- Memiliki dua properti privat: `name` (nama hewan) dan `age` (umur hewan), yang tidak bisa diakses langsung dari luar kelas.

Getter dan Setter:

- **Getter** (`getName()` dan `getAge()`) digunakan untuk mengambil nilai dari properti `name` dan `age`.

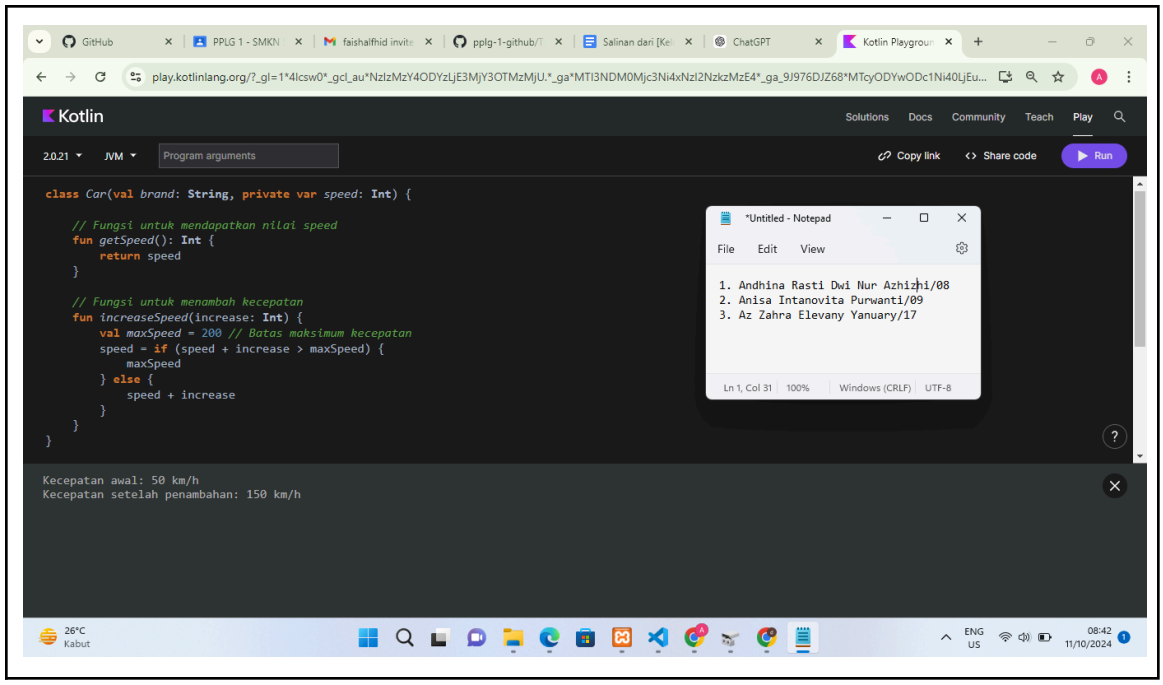
- **Setter** (`setName(newName)` dan `setAge(newAge)`) digunakan untuk mengubah nilai properti `name` dan `age`.

Fungsi `main`:

- Objek `myPet` dibuat dengan nama `"Kucing"` dan umur `2`.
- Getter digunakan untuk mencetak nama dan umur awal.
- Setter digunakan untuk mengubah nama menjadi `"Banteng"` dan umur menjadi `3`, lalu perubahan tersebut dicetak.

Studi kasus berapa kamu? - Nama studi kasusnya apa?

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



2. Lampirkan kode yang kalian buat.

```
class Car(val brand: String, private var speed: Int) {

    // Fungsi untuk mendapatkan nilai speed
    fun getSpeed(): Int {
        return speed
    }

    // Fungsi untuk menambah kecepatan
```

```
    fun increaseSpeed(increase: Int) {
        val maxSpeed = 200 // Batas maksimum kecepatan
        speed = if (speed + increase > maxSpeed) {
            maxSpeed
        } else {
            speed + increase
        }
    }
}

fun main() {
    val car = Car("Toyota", 50)

    // Jack memulai uji coba mobil balap dengan menampilkan kecepatan awal
    println("Kecepatan awal: ${car.getSpeed()} km/h")

    // Jack menambah kecepatan mobil sebanyak 100 km/h
    car.increaseSpeed(100)

    // Tampilkan kecepatan mobil setelah penambahan
    println("Kecepatan setelah penambahan: ${car.getSpeed()} km/h")
}
```

3. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

Kelas `Car`:

- Properti `brand` adalah nama merek mobil dan `speed` adalah kecepatan mobil yang bersifat privat.
- Fungsi `getSpeed()` digunakan untuk mendapatkan kecepatan saat ini.
- Fungsi `increaseSpeed(increase: Int)` menambah kecepatan dengan parameter `increase`, namun memastikan bahwa kecepatan tidak melebihi batas maksimum 200 km/h.

Fungsi `main`:

- Membuat objek `car` dengan merek `"Toyota"` dan kecepatan awal `50 km/h`.
- Menampilkan kecepatan awal mobil.
- Menambah kecepatan mobil sebanyak `100 km/h` menggunakan `increaseSpeed()`.
- Menampilkan kecepatan mobil setelah penambahan.