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

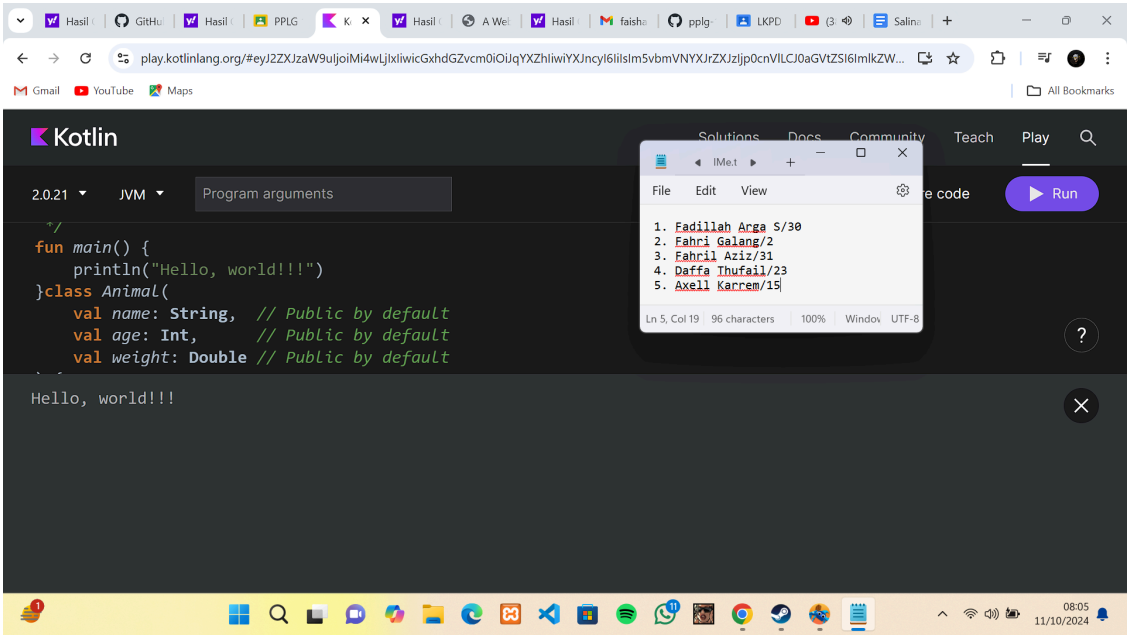
Nama Kelompok : .....

Nama/No.Absen Anggota Kelompok :

1) Fadillah Arga S
2) Fahri Galang
3) Fahril Aziz
4) Daffa Thufail
5) Axell Karrem

TUTORIAL 1 - TUTORIAL PENGGUNAAN VISIBILITY MODIFIERS PUBLIC PADA KOTLIN

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



2. Lampirkan kode yang kalian buat.

```
/**
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */
fun main() {
    println("Hello, world!!!")
}class Animal(
```

```
    val name: String,  // Public by default
    val age: Int,      // Public by default
    val weight: Double // Public by default
) {
    val isMammal: Boolean = true // Public by default
}


    }
```

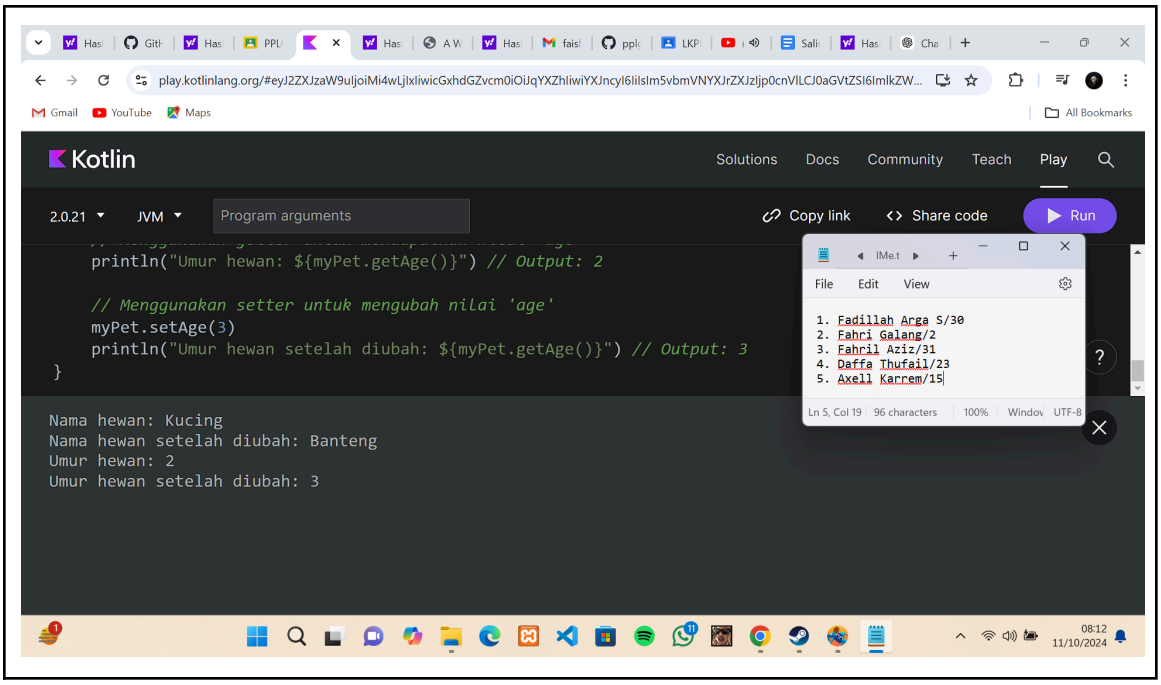
3. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

Kode tersebut ditulis dalam bahasa Kotlin, dimulai dengan fungsi `main()`. Fungsi ini merupakan titik awal eksekusi program. Pada fungsi `main()`, terdapat perintah `println("Hello, world!!!")` yang digunakan untuk mencetak teks "Hello, world!!!" ke layar. Fungsi ini sangat sederhana dan berfungsi untuk menampilkan pesan ke output.

Selanjutnya, terdapat deklarasi kelas `Animal`. Kelas ini memiliki tiga properti, yaitu `name` (tipe data `String`), `age` (tipe data `Int`), dan `weight` (tipe data `Double`). Ketiga properti ini bersifat **public secara default**, artinya dapat diakses dari luar kelas. Selain itu, di dalam kelas `Animal` juga terdapat properti `isMammal`, yang bernilai `true` dan juga bersifat **public**. Properti ini menunjukkan bahwa objek `Animal` secara default adalah mamalia.
.

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
}
```

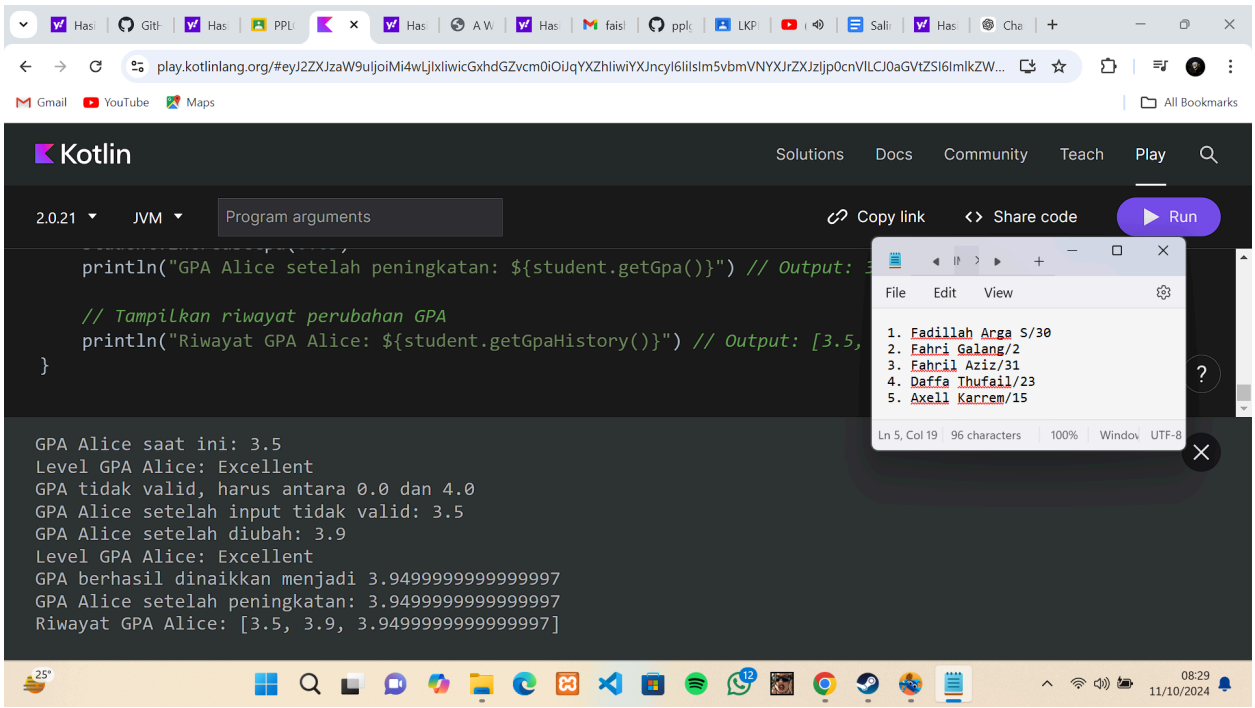
3. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

Kode ini mendefinisikan sebuah kelas `Animal` yang memiliki dua properti privat, yaitu `name` (tipe data `String`) dan `age` (tipe data `Int`). Karena properti tersebut bersifat privat, akses ke properti ini hanya bisa dilakukan melalui metode getter dan setter yang didefinisikan di dalam kelas. Getter `getName()` dan `getAge()` digunakan untuk mengambil nilai `name` dan `age`, sementara setter `setName()` dan `setAge()` digunakan untuk mengubah nilai-nilai tersebut.

Di fungsi `main()`, sebuah objek `Animal` bernama `myPet` diciptakan dengan nilai awal "Kucing" untuk nama dan 2 untuk umur. Program kemudian menggunakan getter untuk mencetak nama dan umur hewan, lalu menggunakan setter untuk mengubah nama menjadi "Banteng" dan umur menjadi 3, dan mencetak hasil perubahannya.

Studi kasus berapa kamu? - Nama studi kasusnya apa?

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



Lampirkan kode yang kalian buat.

```
class Student(val name: String, private var gpa: Double) {

    private val gpaHistory = mutableListOf<Double>() // Untuk menyimpan riwayat GPA

    init {
        if (name.isBlank()) {
            throw IllegalArgumentException("Nama tidak boleh kosong!")
        }
        if (gpa !in 0.0..4.0) {
            throw IllegalArgumentException("GPA harus antara 0.0 dan 4.0")
        }
        gpaHistory.add(gpa)
    }

    // Fungsi untuk mendapatkan nilai gpa
    fun getGpa(): Double {
        return gpa
    }

    // Fungsi untuk mengatur nilai gpa dengan validasi
    fun setGpa(newGpa: Double) {
        if (newGpa in 0.0..4.0) {
            gpa = newGpa
            gpaHistory.add(newGpa)
```

```
        } else {
            println("GPA tidak valid, harus antara 0.0 dan 4.0")
        }
    }

    // Fungsi untuk meningkatkan GPA secara bertahap
    fun increaseGpa(increaseAmount: Double) {
        val newGpa = gpa + increaseAmount
        if (newGpa in 0.0..4.0) {
            setGpa(newGpa)
            println("GPA berhasil dinaikkan menjadi $newGpa")
        } else {
            println("Peningkatan GPA melebihi batas maksimal (4.0). Tidak dapat
meningkatkan GPA.")
        }
    }

    // Fungsi untuk menampilkan deskripsi level GPA
    fun getGpaLevel(): String {
        return when {
            gpa >= 3.5 -> "Excellent"
            gpa >= 3.0 -> "Good"
            gpa >= 2.0 -> "Average"
            else -> "Poor"
        }
    }

    // Fungsi untuk menampilkan riwayat perubahan GPA
    fun getGpaHistory(): List<Double> {
        return gpaHistory
    }
}

fun main() {
    val student = Student("Alice", 3.5)

    // Pak Dedi ingin melihat GPA Alice saat ini
    println("GPA Alice saat ini: ${student.getGpa()}") // Output: 3.5
    println("Level GPA Alice: ${student.getGpaLevel()}") // Output: Excellent

    // Pak Dedi mencoba memasukkan nilai GPA yang tidak valid (contoh: 5.0)
    student.setGpa(5.0) // Output: GPA tidak valid, harus antara 0.0 dan 4.0

    // Lihat apakah nilai GPA berubah setelah input tidak valid
    println("GPA Alice setelah input tidak valid: ${student.getGpa()}") // Output: 3.5
```

```
    // Pak Dedi memperbarui GPA Alice ke nilai yang valid (contoh: 3.9)
    student.setGpa(3.9)
    println("GPA Alice setelah diubah: ${student.getGpa()}") // Output: 3.9
    println("Level GPA Alice: ${student.getGpaLevel()}") // Output: Excellent

    // Pak Dedi meningkatkan GPA Alice
    student.increaseGpa(0.05)
    println("GPA Alice setelah peningkatan: ${student.getGpa()}") // Output: 3.95

    // Tampilkan riwayat perubahan GPA
    println("Riwayat GPA Alice: ${student.getGpaHistory()}") // Output: [3.5, 3.9, 3.95]
}
```

2. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

Kode tersebut mendefinisikan kelas `Student` yang merepresentasikan mahasiswa dengan properti `name` dan `gpa`. Properti `gpa` bersifat privat dan hanya dapat diakses melalui metode getter dan setter, di mana setter memiliki validasi untuk memastikan GPA berada dalam rentang 0.0 hingga 4.0. Selain itu, kode ini juga menyediakan fitur tambahan seperti fungsi untuk menaikkan GPA (`increaseGpa()`), mendapatkan deskripsi level GPA seperti "Excellent" atau "Good" (`getGpaLevel()`), serta mencatat riwayat perubahan GPA melalui list `gpaHistory`. Pada fungsi `main()`, objek `Student` dibuat, dan contoh penggunaan getter, setter, peningkatan GPA, serta riwayat GPA ditampilkan.