Lembar Kerja Peserta Didik  
Modul 4 - OOP Lanjutan

short line

# 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.**

# 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)**.

# PETUNJUK PENGUMPULAN

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

# REFERENSI TAMBAHAN UNTUK BELAJAR

1. Kalian bisa melihat video dari YouTube berikut apabila kalian masih belum paham sepenuhnya mengenai materi yang dipelajari sebelumnya:
2. Penjelasan *visibility modifiers* → [Belajar Kotlin OOP - 27 Visibility Modifier](https://youtu.be/AgQkSWoH7mA?si=vr1SGys40K-r6UZ2)
3. Tutorial dasar *visibility modifiers*  → [Tutorial Kotlin Dasar - 18. Visibility Modifiers](https://youtu.be/sQg40ppMhzc?si=xpWMjBn8fLrt6vl2)

# SOAL

Nama Kelompok : Kylian Mbappe…………………………………..

Nama/No.Absen Anggota Kelompok :

1. …Aldo Ramadani
2. …Ataya Fikri
3. …Amir Mubien
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!

|  |
| --- |

1. 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("Gajah", 2)  // Menggunakan getter untuk mendapatkan nilai 'name'  println("Nama hewan: ${myPet.getName()}") // Output: Kucing  // Menggunakan setter untuk mengubah nilai 'name'  myPet.setName("Cicak")  println("Nama hewan setelah diubah: ${myPet.getName()}") // Output: Banteng  // Menggunakan getter untu\an nilai 'age'  println("Umur hewan: ${myPet.getAge()}") // Output: 2  // Menggunakan setter untuk mengubah nilai 'age'  myPet.setAge(70)  println("Umur hewan setelah diubah: ${myPet.getAge()}") // Output: 3  } |
| --- |

1. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

| Kode di atas mendefinisikan kelas `Animal` yang memiliki dua properti privat: `name` (nama) dan `age` (umur). Kelas ini dilengkapi dengan getter dan setter untuk mengakses dan memodifikasi kedua properti tersebut dari luar kelas.  Penjelasan singkat:  1. \*\*Getter\*\* adalah fungsi untuk mengambil nilai properti. Kode memiliki `getName()` untuk mendapatkan nama hewan dan `getAge()` untuk mendapatkan umurnya.  2. \*\*Setter\*\* adalah fungsi untuk mengubah nilai properti. Kode memiliki `setName(newName)` untuk mengubah nama hewan dan `setAge(newAge)` untuk mengubah umur hewan.  Pada fungsi `main()`, objek `myPet` dari kelas `Animal` dibuat dengan nilai awal `"Gajah"` untuk nama dan `2` untuk umur.  - Pertama, nama hewan diambil menggunakan `getName()` dan dicetak.  - Kemudian nama diubah menjadi `"Cicak"` dengan `setName()`, lalu nama baru dicetak.  - Umur hewan juga diambil menggunakan `getAge()`, kemudian diubah menjadi `70` dengan `setAge()`, dan umur baru dicetak.  Namun, komentar dalam kode kurang tepat, seperti di bagian `// Output: Kucing` dan `// Output: Banteng`, yang seharusnya disesuaikan dengan hasil sebenarnya.  rentang angka tersebut.  **Notes:** Penjelasan hanya contoh, jadi bisa kalian hapus ya. |
| --- |

**TUTORIAL 2 - TUTORIAL PENGGUNAAN VISIBILITY MODIFIERS PRIVATE PADA KOTLIN**

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

|  |
| --- |

1. 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("Aligator", 10, 500.0)  // Mengakses dan memodifikasi properti publik  println("Animal Name: ${elephant.name}") // Output: Elephant  elephant.name = "Big Aligator" // Modifying the name  println("Updated Name: ${elephant.name}") // Output: Big Elephant  } |
| --- |

1. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

| Kode di atas mendefinisikan kelas `Animal` dengan tiga properti publik yang dapat diubah (mutable): `name`, `age`, dan `weight`. Selain itu, ada properti tambahan `isMammal` yang diinisialisasi dengan nilai default `true`.  Penjelasan singkat:  1. \*\*Properti Mutable\*\*: `name`, `age`, dan `weight` adalah properti yang dapat diubah langsung dari luar kelas karena ditandai dengan `var`.  2. \*\*Properti `isMammal`\*\*: Diinisialisasi dengan nilai default `true` dan tidak diinisialisasi melalui konstruktor.  Dalam fungsi `main()`:  - Sebuah objek `elephant` dari kelas `Animal` dibuat dengan nilai `"Aligator"` sebagai nama, `10` sebagai umur, dan `500.0` sebagai berat.  - Properti `name` diakses dan ditampilkan dengan menggunakan `println()`, yang mencetak `"Animal Name: Aligator"`.  - Nilai `name` kemudian diubah menjadi `"Big Aligator"` dan ditampilkan kembali setelah modifikasi.  Catatan: Komentar dalam kode seharusnya diperbaiki, seperti `// Output: Elephant` dan `// Output: Big Elephant`, yang sebenarnya mencetak `"Aligator"` dan `"Big Aligator"`. |
| --- |

**Studi kasus berapa kamu? - Nama studi kasusnya apa?**

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

|  |
| --- |

1. Lampirkan kode yang kalian buat.

| class Phone(private var volume: Int) {  // Fungsi untuk menaikkan volume  fun increaseVolume(amount: Int) {  volume += amount  if (volume > 100) {  volume = 100 // Batas maksimal 100  }  }  // Fungsi untuk menurunkan volume  fun decreaseVolume(amount: Int) {  volume -= amount  if (volume < 0) {  volume = 0 // Batas minimal 0  }  }  // Fungsi untuk mendapatkan volume saat ini  fun getVolume(): Int {  return volume  }  }  // Fungsi main yang benar tanpa argumen  fun main() {  val myPhone = Phone(50) // Volume awal 50    // Tampilkan volume awal  println("Volume awal: ${myPhone.getVolume()}")  // Naikkan volume sebesar 30  myPhone.increaseVolume(30)  println("Volume setelah naik: ${myPhone.getVolume()}")  // Turunkan volume sebesar 70  myPhone.decreaseVolume(70)  println("Volume setelah turun: ${myPhone.getVolume()}")  // Naikkan volume melewati batas maksimal  myPhone.increaseVolume(150)  println("Volume akhir setelah mencoba naikkan melebihi batas: ${myPhone.getVolume()}")  } |
| --- |

1. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

| Berikut penjelasan singkat cara kerja kode:  1. Kelas `Phone`:  - Kelas ini memiliki properti `volume` yang bersifat private, artinya hanya bisa diakses melalui metode di dalam kelas tersebut. Volume diatur dalam rentang 0 hingga 100.  - Metode `increaseVolume(amount: Int)`: Menaikkan volume sesuai jumlah yang diberikan (`amount`). Jika volume yang baru lebih besar dari 100, maka volume akan diatur ke 100 (batas maksimum).  - Metode `decreaseVolume(amount: Int)`: Menurunkan volume sesuai jumlah yang diberikan (`amount`). Jika volume yang baru kurang dari 0, maka volume akan diatur ke 0 (batas minimum).  - Metode `getVolume()`: Mengembalikan nilai volume saat ini.  2. \*\*Fungsi `main()`:  - Membuat objek `Phone` dengan volume awal 50.  - Menampilkan volume awal menggunakan metode `getVolume()`.  - Menaikkan volume sebesar 30, kemudian menampilkan hasilnya.  - Menurunkan volume sebesar 70, kemudian menampilkan hasilnya.  - Mencoba menaikkan volume sebesar 150, lalu menampilkan volume yang seharusnya tetap di batas maksimal 100.  Kesimpulan:  Program ini memastikan volume ponsel tidak pernah melebihi 100 atau kurang dari 0 saat dinaikkan atau diturunkan, dan memberikan output nilai volume saat ini setiap kali volume diubah. |
| --- |