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 :Serigala kejam 77

Nama/No.Absen Anggota Kelompok :

1. Christian Akasa
2. Dunda Arya
3. Bagus Pandu
4. Aurel Messi
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. |
| --- |

1. Lampirkan kode yang kalian buat.

| class Animal(  var nama: String,  var umur: Int,  var weight: Double  ) {  var isMammal: Boolean = true  }  fun main() {  val Jerapah = Animal("Giraffe", 6, 200.0)    println("Animal Name: ${Jerapah.nama}")  Jerapah.nama = "Small Girrafe"  println("Updated Name: ${Jerapah.nama}")  }  **Notes:** Kode hanya contoh, jadi bisa kalian hapus ya. |
| --- |

1. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

| Kode tersebut merupakan program yang mencetak bilangan ganjil dari 1 hingga 1000. Perulangan for digunakan untuk mengiterasi setiap bilangan dari 1 hingga 1000, dengan variabel i sebagai penampung setiap nilai dalam rentang tersebut. Di dalam perulangan, terdapat kondisi if (i % 2 == 0) yang digunakan untuk memeriksa apakah nilai i merupakan bilangan genap. Jika i genap, perintah continue akan dijalankan, yang menyebabkan program melompati iterasi tersebut tanpa mencetak apa pun. Sebaliknya, jika i adalah bilangan ganjil, perintah println(i) akan dijalankan untuk mencetak nilai i. Dengan demikian, hanya bilangan ganjil yang akan dicetak dalam 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.

| /\*\*  \* You can edit, run, and share this code.  \* play.kotlinlang.org  \*/    class Animal(private var name: String, private var age: Int) {    fun getName(): String {  return name  }  fun setName(newName: String) {  name = newName  }    fun getAge(): Int {  return age  }  fun setAge(newAge: Int) {  age = newAge  }  }  fun main() {  val myPet = Animal("Anjing", 3)    println("Nama hewan: ${myPet.getName()}")    myPet.setName("Anoa")  println("Nama hewan setelah diubah: ${myPet.getName()}")    println("Umur hewan: ${myPet.getAge()}")    myPet.setAge(4)  println("Umur hewan setelah diubah: ${myPet.getAge()}") // Output: 3  } |
| --- |

1. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

| Kode tersebut mendefinisikan kelas `Animal` dengan dua properti: `name` dan `age`, yang bersifat privat. Ada metode getter dan setter untuk mengakses dan mengubah nilai properti tersebut. Di dalam `main()`, objek `myPet` dibuat dengan nama "Anjing" dan umur 3. Kemudian, nama dan umur hewan diubah menggunakan metode setter, dan hasilnya ditampilkan dengan `println`. |
| --- |

**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 Book(val title: String, private var pages: Int) {  // Fungsi untuk mendapatkan jumlah halaman  fun getPages(): Int {  return pages  }  // Fungsi untuk mengatur jumlah halaman dengan validasi  fun setPages(newPages: Int) {  if (newPages > 0) {  pages = newPages  } else {  println("Jumlah halaman tidak valid!")  }  }  }  fun main() {  val book = Book("Pemrograman Kotlin", 300)    // Penerbit ingin melihat jumlah halaman awal  println("Jumlah halaman awal: ${book.getPages()}")    // Penerbit mencoba mengubah jumlah halaman menjadi nilai yang tidak valid (contoh: -10)  book.setPages(-10)    // Tampilkan jumlah halaman setelah perubahan tidak valid  println("Jumlah halaman setelah perubahan tidak valid: ${book.getPages()}")    // Penerbit memperbarui jumlah halaman menjadi nilai yang valid (contoh: 350)  book.setPages(350)    // Tampilkan jumlah halaman setelah perubahan valid  println("Jumlah halaman setelah perubahan valid: ${book.getPages()}")  } |
| --- |

1. Selanjutnya jelaskan secara singkat cara kerja kode kalian!

| Kode tersebut membuat kelas `Book` dengan dua properti: `title` (judul) dan `pages` (jumlah halaman). Properti `pages` bersifat privat dan hanya bisa diakses serta diubah melalui metode getter dan setter.  1. \*\*Getter (`getPages()`)\*\*: Mengembalikan nilai jumlah halaman.  2. \*\*Setter (`setPages(newPages: Int)`)\*\*: Mengubah jumlah halaman, namun hanya jika nilai yang diberikan lebih besar dari 0. Jika tidak, akan muncul pesan "Jumlah halaman tidak valid!" dan jumlah halaman tidak berubah.  Di dalam `main()`, objek `book` dibuat dengan judul "Pemrograman Kotlin" dan jumlah halaman 300. Program mencoba mengubah jumlah halaman ke -10 (tidak valid) dan kemudian ke 350 (valid), lalu mencetak hasilnya.  ### Hasil:  - Jika set halaman dengan -10 (tidak valid), jumlah halaman tetap 300.  - Jika set halaman dengan 350 (valid), jumlah halaman berubah menjadi 350. |
| --- |