

The screenshot shows the Android Studio interface with the code editor open to the `Aktor.java` file. The code defines an abstract class `Aktor` with protected attributes `idAktor`, `nama`, `alamat`, and `noHp`. It includes a constructor and an abstract method `tampilkanMenu()`.

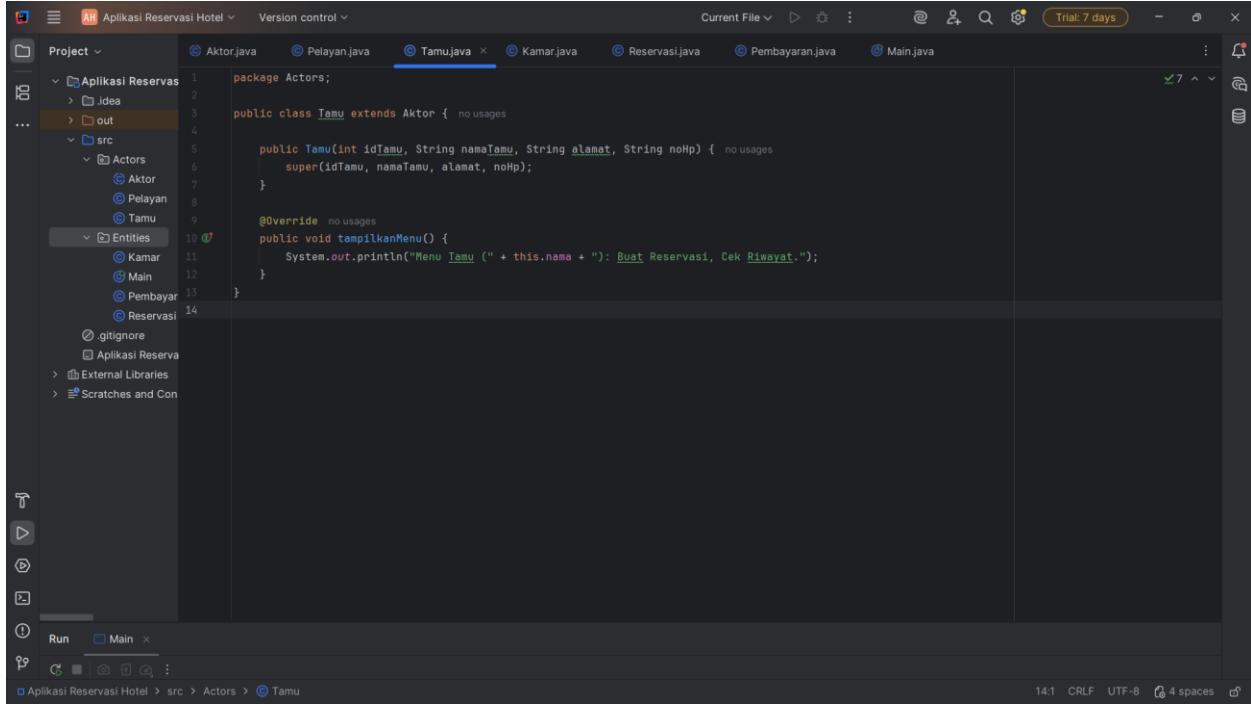
```
package Actors;
public abstract class Aktor {
    protected int idAktor;
    protected String nama;
    protected String alamat;
    protected String noHp;
    public Aktor(int idAktor, String nama, String alamat, String noHp) {
        this.idAktor = idAktor;
        this.nama = nama;
        this.alamat = alamat;
        this.noHp = noHp;
    }
    public abstract void tampilkanMenu();
}
```

Kelas **Aktor** untuk dipakai menyimpan data orang yang terlibat di sistem, yaitu pelayan dan tamu. Di dalamnya ada informasi seperti ID, nama, alamat, dan nomor HP. Kelas ini dibuat sebagai kelas abstrak, jadi nggak bisa dipakai langsung. Tujuannya supaya kelas turunannya punya aturan wajib untuk membuat fungsi `tampilkanMenu()`. Intinya, `Aktor` ini jadi pondasi untuk class `Pelayan` dan `Tamu`.

The screenshot shows the Android Studio interface with the code editor open to the `Pelayan.java` file. The code extends the `Aktor` class and overrides the `tampilkanMenu()` method to print a specific message.

```
package Actors;
public class Pelayan extends Aktor {
    public Pelayan(int idPelayan, String namaPelayan, String alamat, String noHp) {
        super(idPelayan, namaPelayan, alamat, noHp);
    }
    @Override
    public void tampilkanMenu() {
        System.out.println("Menu Pelayan (" + this.nama + "): Input Reservasi, Proses Pembayaran.");
    }
}
```

Kelas **Pelayan** dari Aktor dan mewakili pegawai hotel. Di sini pelayan punya menu yang khusus, seperti memasukkan reservasi atau mengurus pembayaran. Kelas ini mengubah metode *tampilkanMenu()* supaya apa yang muncul sesuai dengan tugas pelayan. Data identitas pelayan juga diwariskan dari class Aktor.



```
package Actors;

public class Tamu extends Actor {    no usages

    public Tamu(int idTamu, String namaTamu, String alamat, String noHp) {        no usages
        super(idTamu, namaTamu, alamat, noHp);
    }

    @Override    no usages
    public void tampilkanMenu() {
        System.out.println("Menu Tamu (" + this.nama + "): Buat Reservasi, Cek Riwayat.");
    }
}
```

Kelas **Tamu** turunan dari Aktor. Bedanya, menu yang ditampilkan sesuai kebutuhan tamu, misalnya membuat reservasi dan melihat riwayat. Sama seperti Pelayan, kelas ini juga menimpa metode *tampilkanMenu()* supaya tampilannya cocok untuk tamu. Intinya ini contoh penggunaan *inheritance* dan *polymorphism* di program.

```

package Entities;

public class Kamar {
    private int idKamar;
    private String tipeKamar;
    private int harga;
    private String status;

    public Kamar(int idKamar, String tipeKamar, int harga) {
        this.idKamar = idKamar;
        this.tipeKamar = tipeKamar;
        this.harga = harga;
        this.status = "Tersedia";
    }

    public int getIdKamar() {
        return idKamar;
    }

    public int getHarga() {
        return harga;
    }

    public void ubahStatus(String statusBaru) {
        this.status = statusBaru;
        System.out.println("[Kamar " + idKamar + "] Status diubah menjadi: " + statusBaru);
    }
}

```

Kelas **Kamar** untuk menyimpan informasi kamar hotel. Tiap kamar punya ID, tipe kamar, harga, dan status. Status awalnya selalu “Tersedia”, dan bisa diubah lewat metode *ubahStatus()*. Kelas ini nanti bakal dipakai saat melakukan proses reservasi untuk menentukan apakah kamar kosong atau sudah terisi.

```

package Entities;

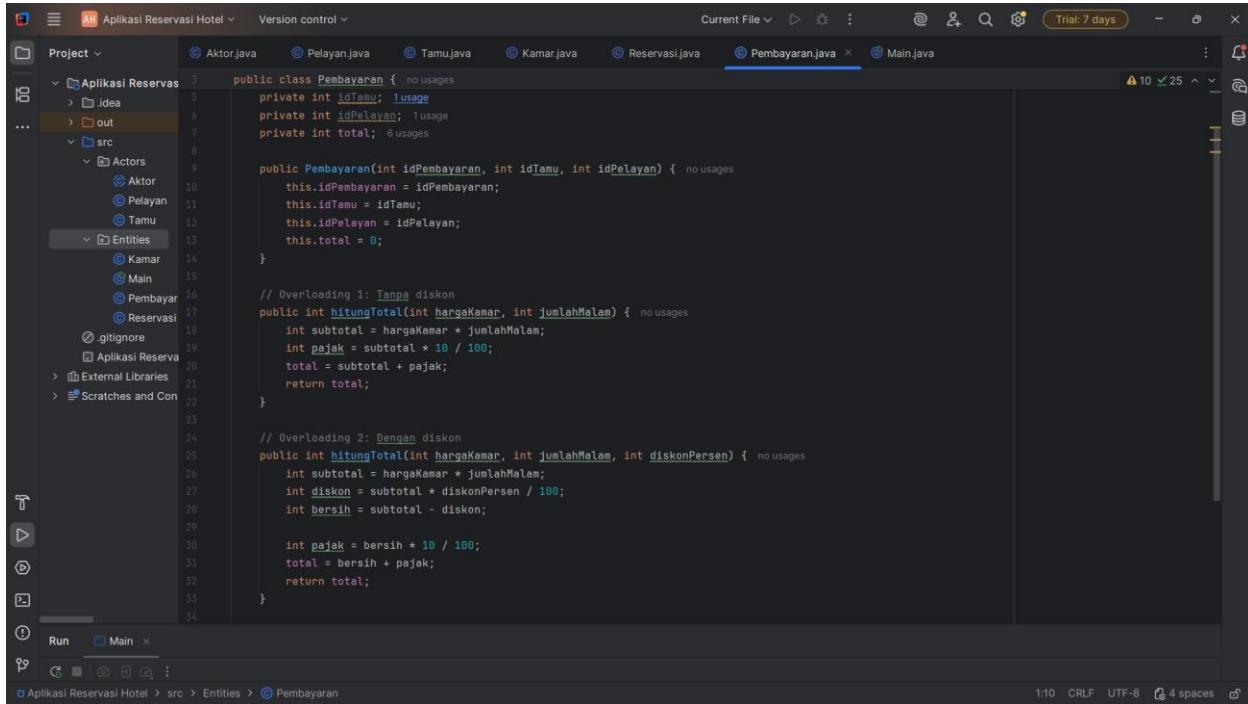
public class Reservasi {
    private int idReservasi;
    private int idPelayan;
    private int idKamar;
    private int idTamu;

    public Reservasi(int idReservasi, int idPelayan, int idKamar, int idTamu) {
        this.idReservasi = idReservasi;
        this.idPelayan = idPelayan;
        this.idKamar = idKamar;
        this.idTamu = idTamu;
    }

    public void buatReservasi(Kamar kamar) {
        kamar.ubahStatus("Terisi");
        System.out.println("[Reservasi " + idReservasi + "] Berhasil dibuat oleh Pelayan "
            + idPelayan + " untuk Tamu " + idTamu + ".");
    }
}

```

Kelas **Reservasi** menangani proses pemesanan kamar. Di sini ada data siapa pelayannya, siapa tamunya, kamar yang dipakai, serta ID reservasinya. Ada metode *buatReservasi()* yang tugasnya mengubah status kamar jadi “Terisi” dan menampilkan info kalau reservasi sudah berhasil dibuat. Kelas ini jadi penghubung antara tamu, pelayan, dan kamar.



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "Aplikasi Reservas". The "src" folder contains packages like "Actors", "Entities", and "Reservasi".
- Code Editor:** Displays the `Pembayaran.java` file. The code defines a class `Pembayaran` with two methods: `hitungTotal` (overloaded) and `hitungTotal` (overloaded).
- Toolbars and Status Bar:** Includes standard IntelliJ IDEA icons and status information like "Trial: 7 days", "10 25", "Run Main", "1:10 CRLF UTF-8 4 spaces", and "Scratches and Con".

```
public class Pembayaran {    private int idTamu;    private int idPelayan;    private int total;    public Pembayaran(int idPembayaran, int idTamu, int idPelayan) {        this.idPembayaran = idPembayaran;        this.idTamu = idTamu;        this.idPelayan = idPelayan;        this.total = 0;    }    // Overloading 1: Tanpa diskon    public int hitungTotal(int hargaKamar, int jumlahMalam) {        int subtotal = hargaKamar * jumlahMalam;        int pajak = subtotal * 10 / 100;        total = subtotal + pajak;        return total;    }    // Overloading 2: Dengan diskon    public int hitungTotal(int hargaKamar, int jumlahMalam, int diskonPersen) {        int subtotal = hargaKamar * jumlahMalam;        int diskon = subtotal * diskonPersen / 100;        int bersih = subtotal - diskon;        int pajak = bersih * 10 / 100;        total = bersih + pajak;        return total;    }}
```

Kelas **Pembayaran** dipakai untuk menghitung total biaya menginap. Di sini ada ID pembayaran, ID tamu, ID pelayan, dan total harga. Yang menarik, kelas ini punya dua versi fungsi *hitungTotal()*:

- satu tanpa diskon
- satu lagi dengan diskon

Ini contoh penggunaan *method overloading* dalam Java. Hasil akhir harga juga ditambahkan pajak 10%.

The screenshot shows a Java development environment with the following details:

- Project Structure:** The project is named "Aplikasi Reservas". It contains a package named "Entities" which includes classes for "Kamar", "Main", "Pembayaran", and "Reservasi".
- Main.java Content:** The code demonstrates inheritance and overriding. It prints a welcome message, creates instances of Pelayan, Tamu, and Kamar, and then prints menu lists for each.
- Code Snippet:**

```
import Actors.Pelayan;
import Actors.Tamu;
import Entities.Kamar;
import Entities.Reservasi;
import Entities.Pembayaran;

public class Main {
    public static void main(String[] args) {
        System.out.println("---- DEMO APLIKASI RESERVASI HOTEL ----");

        Pelayan pelayan1 = new Pelayan( idPelayan: 101, namaPelayan: "Aulia", alamat: "Jl. Mawar 50", noHp: "081234567890");
        Tamu tamu1 = new Tamu( idTamu: 201, namaTamu: "Bambang", alamat: "JL. Melati 10", noHp: "089876543210");
        Kamar kamar101 = new Kamar( idKamar: 101, tipeKamar: "STD", harga: 300000);

        System.out.println("\n### 1. Demonstrasai Inheritance & Overriding ###");
        pelayan1.tampilkanMenu();
        tamu1.tampilkanMenu();

        System.out.println("\n### 2. Demonstrasai Relasi Reservasi ###");
        Reservasi reservasiBaru = new Reservasi(
            idReservasi: 9001,
            idPelayan: 101,
            kamar101.getIdKamar(),
            idTamu: 201
        );
        reservasiBaru.buatReservasi(kamar101);

        System.out.println("\n### 3. Demonstrasai Overloading ###");
        Pembayaran pembayaran = new Pembayaran( idPembayaran: 8001, idTamu: 201, idPelayan: 101);
```

Dan ini MAIN nya