Nama: Aqsha Ramadhan Dimas Haryanto
NIM: H1D024089
SHIFT BARU: D
LAPORAN RESPONSI 2 PBO PERTEMUANl: 6

A. Alur kerja
   1. Inisialisasi Objek (Constructor Chaining)
- Program membuat objek astra (PesawatTempur) dan voyager (KapalEksplorasi).
- Saat new PesawatTempur(...) dipanggil, ia menjalankan baris super(...) terlebih dahulu.
- Ini mengirim data Nama dan Kapasitas ke constructor induk (KendaraanGalaksi) untuk disimpan, sekaligus mengatur energi awal otomatis menjadi 100%.

   2. Eksekusi Method Abstrak (Polimorfisme)
- Aktifkan Mesin:
    - astra.aktifkanMesin() mengecek apakah energi > 20.
    - voyager.aktifkanMesin() mengecek apakah energi > 15.
    - *Poin Penting:* Meskipun nama methodnya sama, aturan (logika)-nya berbeda tergantung jenis kendaraannya.
- Penjelajahan (Jelajah):
    - astra berjalan 10 km $\rightarrow$ Energi berkurang $10 \times 3 = 30\%$.
    - voyager berjalan 15 km $\rightarrow$ Energi berkurang $15 \times 2 = 30\%$.
    - Jika energi tidak cukup (seperti saat astra mencoba jalan 30 km lagi), sistem menolak perintah tersebut.

   3. Eksekusi Fitur Unik (Spesialisasi)
- Program memanggil method yang hanya dimiliki oleh anak tertentu:
    - astra.tembakRudal() $\rightarrow$ Mengurangi jumlah amunisi rudal.
    - voyager.scanPlanet() $\rightarrow$ Menampilkan simulasi scanning planet.

   4. Pewarisan Method Konkrit (tampilStatus)
- Di akhir, program memanggil astra.tampilStatus() dan voyager.tampilStatus().
- Meski method ini tidak ditulis di dalam file Pesawat/Kapal, program berhasil mencetak info Nama, Energi, dan Kapasitas.
- Ini terjadi karena method tersebut diwariskan dari KendaraanGalaksi

B. Fungsi yang digunakan
   1. Abstract Class (abstract class KendaraanGalaksi)
- Fungsi: Sebagai *blueprint* (cetakan dasar) yang belum sempurna. Class ini tidak bisa dibuat objeknya secara langsung (new KendaraanGalaksi = Error).
- Tujuan: Menjamin bahwa semua kendaraan galaksi PASTI punya nama, energi, dan kapasitas, tapi membiarkan detail cara kerjanya ditentukan nanti oleh anak-anaknya.

   2. Abstract Method (abstract void ...)
- Implementasi: aktifkanMesin(), jelajah(), isiEnergi().
- Fungsi: Memaksa class anak (PesawatTempur & KapalEksplorasi) untuk membuat kode logika method tersebut. Jika anak tidak membuatnya, program akan error.
- Kenapa dipakai? Karena cara pesawat tempur terbang (boros) beda dengan kapal eksplorasi (hemat). Induk tidak tahu caranya, jadi dia menyerahkan tanggung jawab itu ke anak.

3. Inheritance (extends)

- Fungsi: Mewariskan atribut (nama, energi) dan method biasa (tampilStatus) dari Induk ke Anak.
- Manfaat: Efisiensi kode. Kita tidak perlu menulis ulang method tampilStatus di setiap kendaraan baru.

4. Encapsulation (private & protected)

- Implementasi: Atribut di induk bersifat private, tapi kita menyediakan method protected void setLevelEnergi().
- Fungsi: Keamanan data.
    - private: Agar energi tidak bisa diubah sembarangan dari luar (misal di-hack jadi 1000%).
    - protected setter: Memberikan izin khusus hanya kepada class Anak (Pesawat/Kapal) untuk mengurangi energi saat menjelajah.

5. Keyword final pada Method

- Implementasi: public final void tampilStatus().
- Fungsi: Mengunci method tersebut agar tidak bisa diubah (override) oleh class anak. Ini menjaga agar format laporan status kendaraan tetap seragam di seluruh sistem galaksi

Hasil Output Program:

```
[Running] cd "d:\Responsi-2\pert6\" && javac UjiGalaksi_.java && java UjiGalaksi_
=== UJI SISTEM KENDARAAN GALAKSI ===

--- PESAWAT TEMPUR ---
Mesin pesawat tempur diaktifkan.
Pesawat tempur menjelajah sejauh 10 km.
Energi tidak mencukupi untuk menjelajah sejauh 30 km.
Menembakkan 3 rudal!
Astra-Fury | Energi: 70% | Kapasitas: 2 awak

--- KAPAL EKSPLORASI ---
Kapal eksplorasi siap berangkat!
Kapal eksplorasi menjelajah sejauh 15 km.
Melakukan scan pada planet Kepler-442b dengan modul level 4.
Voyager X | Energi: 70% | Kapasitas: 10 awak
```