

LAPORAN PERTEMUAN 6

NAMA : AHMAD FIKRI ZAKARIA
NIM : H1D024062
MATERI : ABSTRACT CLASS & METHOD

A. Alur Kerja (Workflow)

Pada pertemuan ini, saya membangun sistem simulasi kendaraan antariksa menggunakan konsep *Abstract Class*. Tujuannya adalah membuat standarisasi perilaku kendaraan namun membiarkan implementasi teknisnya berbeda. Langkah-langkah pengerjaannya meliputi:

1. **Pembuatan Abstract Class (KendaraanGalaksi):**
 - Mendefinisikan atribut private (nama, energi, kapasitas) beserta *Getter/Setter*.
 - Membuat *constructor* untuk set nilai awal energi 100%.
 - Mendefinisikan tiga **Abstract Method**: *aktifkanMesin()*, *jelajah(int)*, dan *isiEnergi(int)* yang wajib diimplementasikan ulang oleh anak kelas.
2. **Pembuatan Concrete Subclass (PesawatTempur):**
 - Mewarisi *KendaraanGalaksi*.
 - Mengimplementasikan logika boros energi (3% per km) pada method *jelajah*.
 - Menambahkan fitur khusus *tembakRudal*.
3. **Pembuatan Concrete Subclass (KapalEksplorasi):**
 - Mewarisi *KendaraanGalaksi*.
 - Mengimplementasikan logika hemat energi (2% per km) pada method *jelajah*.
 - Menambahkan fitur khusus *scanPlanet*.
4. **Pengujian (UjiGalaksi):**
 - Membuat objek pesawat dan kapal, lalu menguji batas energi dan fitur unik masing-masing untuk memastikan polimorfisme dan abstraksi berjalan benar.

B. Penjelasan Kode & Fungsi

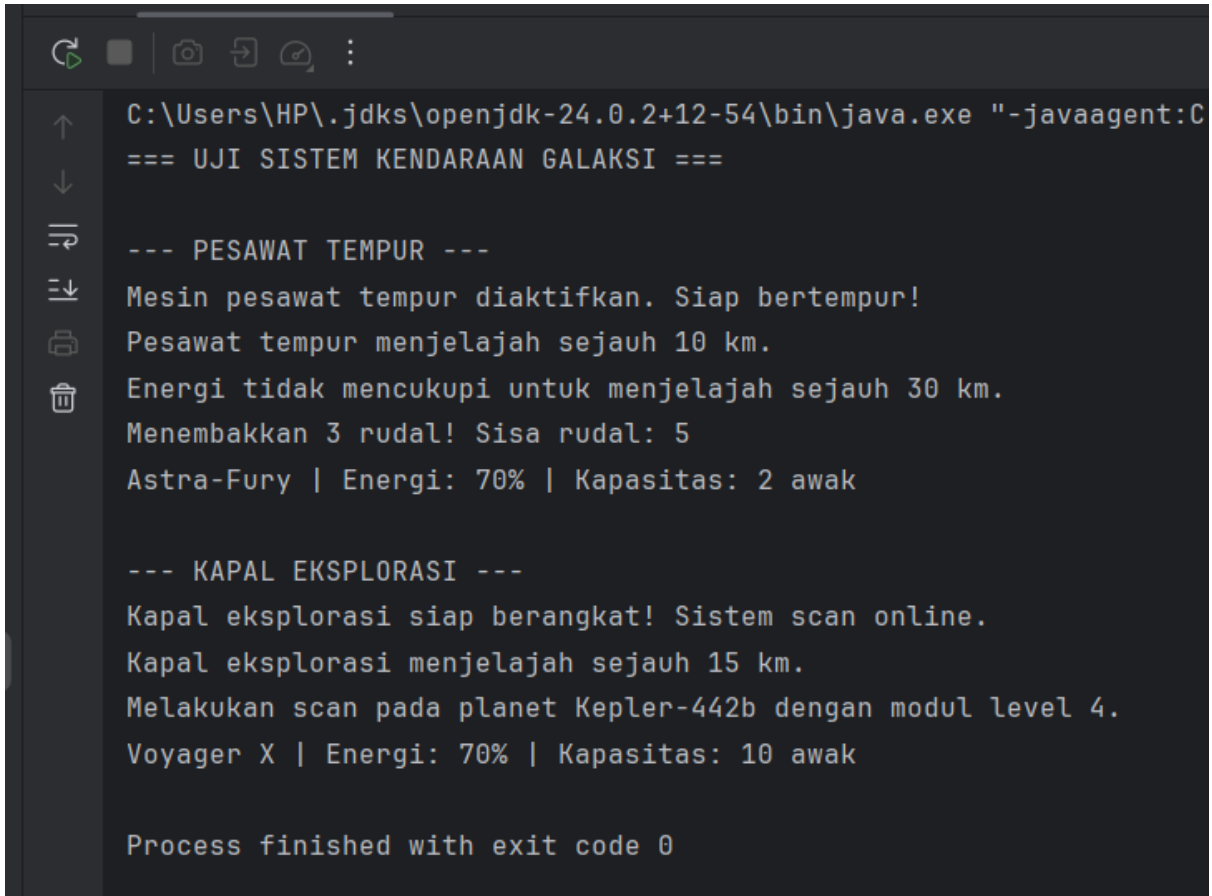
Berikut adalah analisis mendalam mengenai konsep teknis yang diterapkan dalam kode program:

1. **Abstract Class & Method (abstract)** Class *KendaraanGalaksi* didefinisikan sebagai *abstract class*, yang berarti class ini tidak dapat diinstansiasi secara langsung menjadi objek (tidak bisa *new KendaraanGalaksi*). Ia berfungsi sebagai kerangka dasar (*blueprint*). Di dalamnya terdapat *abstract methods* (*aktifkanMesin*, *jelajah*, *isiEnergi*) yang hanya berupa deklarasi tanpa implementasi (*body*). Hal ini mewajibkan setiap kelas turunannya untuk menyediakan logika konkret masing-masing, menjamin standarisasi struktur namun fleksibilitas dalam perilaku.
2. **Implementation & Overriding** Class *PesawatTempur* dan *KapalEksplorasi* sebagai *concrete subclass* wajib melakukan *override* terhadap semua *abstract method* milik induknya.
 - Pada method *jelajah(int jarak)*, **PesawatTempur** menerapkan logika konsumsi energi tinggi (3% per km), sedangkan **KapalEksplorasi** menerapkan logika efisien (2% per km).
 - Ini menunjukkan bagaimana satu nama method yang sama dapat memiliki alur kerja yang sangat berbeda tergantung pada objek spesifik yang menjalankannya.
3. **Encapsulation pada Inheritance** Atribut seperti *levelEnergi* dan *namaKendaraan* dideklarasikan sebagai private di dalam *abstract class* untuk menjaga keamanan data. Karena atribut private tidak diwariskan secara langsung, *subclass* memanipulasi data tersebut melalui bantuan method *Getter* dan *Setter* (misalnya *setLevelEnergi* dan *getLevelEnergi*) yang

disediakan oleh *superclass*. Ini memastikan bahwa perubahan status energi tetap terkontrol melalui satu pintu.

C. Hasil Output Program

Berikut adalah tangkapan layar/teks hasil eksekusi program UjiPengguna.java:



```
C:\Users\HP\.jdk\openjdk-24.0.2+12-54\bin\java.exe "-javaagent:C
=== UJI SISTEM KENDARAAN GALAKSI ===

--- PESAWAT TEMPUR ---
Mesin pesawat tempur diaktifkan. Siap bertempur!
Pesawat tempur menjelajah sejauh 10 km.
Energi tidak mencukupi untuk menjelajah sejauh 30 km.
Menembakkan 3 rudal! Sisa rudal: 5
Astra-Fury | Energi: 70% | Kapasitas: 2 awak

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

Process finished with exit code 0
```