

Zadanie Projektowe - System Pojazdow (Java OOP)

Celem projektu jest przecwiczenie wszystkich podstawowych elementow programowania obiektowego (OOP) w Javie: enkapsulacji, dziedziczenia, abstrakcji, interfejsow, polimorfizmu oraz slow kluczowych this i final. Projekt bedzie polegal na stworzeniu prostego systemu do zarzadzania pojazdami.

Struktura plikow:



Pojazd.java

- Klasa abstrakcyjna (nie mozna jej uzyc do stworzenia obiektu).
- Zawiera prywatne pola: marka, rokProdukcji.
- Konstruktor uzywajacy this.
- Gettery i settery z walidacja (rok >= 1886).
- Metoda abstrakcyjna obliczSpalanie(int km).
- Metoda final info() - wyswietla dane pojazdu.

Samochod.java

- Dziedziczy po Pojazd.
- Pole: liczbaDrzwi.
- Konstruktor uzywajacy super(...).
- Nadpisana metoda obliczSpalanie(int km).
- Metoda toString() uzywajaca super.toString().

Motocykl.java

- Dziedziczy po Pojazd.
- Pole: czyMaKosz (boolean).
- Konstruktor.
- Implementacja obliczSpalanie(int km).
- Wlasne toString().

Elektryczny.java

- Interfejs z metoda: void naladuj(int procent).

SamochodElektryczny.java

- Dziedziczy po Samochod.
- Implementuje interfejs Elektryczny.
- Pole: poziomBaterii (0-100).
- Implementacja naladuj(int procent).
- Wlasna metoda obliczSpalanie(int km).

Stale.java

- Klasa z polami static final.
- Zawiera np. `public static final double CENA_PALIWA = 6.50;`

Matematyka.java

- Klasa final (nie można jej rozszerzać).
- Metody statyczne np. `dodaj(int a, int b)`.

Main.java

- Zawiera metodę `main()`.
- Tworzy obiekty `Samochod`, `Motocykl`, `SamochodElektryczny`.
- Dodaje je do tablicy `Pojazd[]`.
- W pętli wywołuje `toString()` i `obliczSpalanie(100)`.
- Wywołuje `info()` i `naladuj()`.
- Używa `Matematyka.dodaj(...)`.

Efekt końcowy (przykład w konsoli):

- Samochod: Toyota, rok 2020, drzwi: 4
- Spalanie na 100 km: 7.5 L
-
- Motocykl: Yamaha, rok 2018, koszt: tak
- Spalanie na 100 km: 3.5 L
-
- Samochod elektryczny: Tesla, rok 2022, bateria: 80%
- Zużycie na 100 km: 15 kWh