# *Java: Dziedziczenie, polimorfizm, interfejsy, UML - Ćwiczenia*

## *Ćwiczenie 1*

a) Utwórz abstrakcyjną klasę `models.Vehicle` zawierającą:
   - Prywatne pola: `speed` typu `int` i `model` typu `String`
   - Publiczne metody `setSpeed`, `setModel`, `getSpeed`, `getModel` o odpowiednich argumentach i typach zwrotnych. Metody te możesz zdefiniować ręczenie lub wygenerować za pomocą kreatora (Refactor → Encapsulate Fields….) na podstawie wcześniej zdefiniowanych pól.
   - 2-argumentowy publiczny konstruktor ustawiający `speed` i `model` na podstawie wartości podanych jako parametry wywołania.
   - Publiczną metodę zwracającą `String` zawierający informacje o modelu pojazdu i jego prędkości np. w formacie: „model:XXX speed:YYY”. Nazwij metodę tak aby była wykorzystywana jako domyślna metoda zwracająca tekstową reprezentację obiektu (nadpisz odpowiednią metodę odziedziczoną z `java.lang.Object`).

b) Utwórz klasę `models.Car` dziedziczącą z `Vehicle`:
   - Zawierającą dodatkowo prywatne pole `numberOfDoors` i publiczne metody `getNumberOfDoors` i `setNumberOfDoors`.
   - Zawierającą 3-argumentowy konstruktor ustawiający wartości pól tworzonego obiektu.
   - Nadpisującą publiczną metodę zwracającą tekstową reprezentację obiektu z wywołaniem metody odziedziczonej i dodatkowo uwzględniającą liczbę drzwi.

c) Utwórz klasę `models.Plane` dziedziczącą z `Vehicle`:
   - Zawierającą dodatkowo prywatne pole `numberOfSeats` i publiczne metody `getNumberOfSeats` i `setNnumberOfSeats`.
   - Zawierającą 3-argumentowy konstruktor ustawiający wartości pól tworzonego obiektu.
   - Nadpisującą publiczną metodę zwracającą tekstową reprezentację obiektu z wywołaniem metody odziedziczonej i dodatkowo uwzględniającą liczbę miejsc.

d) Utwórz startową klasę `models.App` i w jej metodzie `main` kolejno:
   - Utwórz tablicę do składowania 4 pojazdów.
   - Wstaw do tablicy 2 samochody i 2 samoloty
   - Przejdź tablicę pętlą `for` i wyświetl opisy pojazdów
   - Przejdź tablicę pętlą “`foreach`” (`for :`) i wyświetl opisy pojazdów.

e) Uruchom program z poziomu środowiska IDE.

### *Ćwiczenie 2*

a) Dodaj do aplikacji interfejs `models.Tuningable` zawierający metodę `increaseSpeed` z parametrem typu `int`.

b) Spraw aby klasa `Car` implementowała ten interfejs. Dodaj w klasie `Car` implementację metody `increaseSpeed`.

c) W metodzie `main` po wypełnieniu tablicy obiektami `Vehicle` zadeklaruj zmienną typu `Tuningable`. Przypisz do niej referencję do jednego z samochodów. Za pomocą zmiennej typu `Tuningable` zwiększ prędkość tego samochodu.

d) Uruchom aplikację.

### *Ćwiczenie 3*

Korzystając z narzędzia UMLet narysuj diagram klas UML obejmujący hierarchię klas pojazdów (`Vehicle`, `Car`, `Plane`) i interfejs `Tuningable`.