Laboratorium 2 i 3. Własności klas. Obiekty. Metody

Celem dzisiejszego laboratorium będzie przypomnienie pojęć klasy, pola, metody, modyfikatorów dostępu oraz tworzenia obiektu poznanych na zajęciach z C++, jak również w zrozumieniu różnic jakie występują w Javie. Materiały pomocnicze są zamieszczone na mojej stronie.

Począwszy od tych zajęć będziemy modelować obiekt samochodu. Zadanie laboratoryjne będzie się składało z poniższych części:

1. Klasa Samochód posiada pola: marka, masa, rok produkcji, kilometraż i pojemność silnika. W pierwszej kolejności ustalamy, jakie typy danych mogą być użyte w polach i odpowiednio je zadeklarowujemy (Jak deklarować zmienne w Javie, nauczyliśmy się na poprzednich laboratoriach). Tu relizujemy również pojęcie modyfikatora dostępu (na razie tylko public i private). Pola marka, pojemność silnika i rok produkcji będą opisywane jako prywatne. **Każde** pole powinno posiadać modyfikator dostępu.

2. Mając opisaną strukturę klasy, możemy przejść do narzędzia, które stworzy na obiekt klasy Samochód – czyli konstruktor. Konstruktor w Javie ma **taką samą nazwę jak klasa** i może być **pusty (bezargumentowy)** albo **z argumentami (parametrami przekazywanymi w nawiasach)**. Konstruktor ma zazwyczaj **publiczny** modyfikator dostępu. W tym zadaniu mamy stworzyć dwa konstruktory: jeden bezargumentowy inicjalizujący domyślnie wszystkie pola klasy (pola masy, kilometrażu i pojemności silnika mogą mieć ustaloną wartość zaś marka jest wybierana przypadkowo spośród tablicy producentów: Audi, VW, BMW, Mercedes, Toyota). Drugi konstruktor powinien być z argumentami, gdzie wymagane będą marka, kilometraż i pojemność silnika (czyli przekazujemy tylko 3 parametry, pozostałe pola powinny być ustawione o wartość domyślną) .

3. Klasa posiadająca prywatne pola powinna mieć metody do pobierania i ustawiania wartości powyższych pól. Dane metody nazywamy potocznie ‚getter' i ‚setter'. Zazwyczaj deklarujemy je tak: **public typ getNazwaPola() {return nazwaPola;} public setNazwaPola(typ parametr) {nazwaPola = parametr;}** W tym zadaniu storzymy getery i settery dla wszystkich prywatnych pól. Dla przynajmniej jednego z nich stworzyć setter w krórym nazwa parametru **jest taka sama jak nazwa pola** ().

4. Oprócz getterów i setterów, klasa może posiadać inne metody. Stwórzmy więc publiczną metodę **jedź()** mającą argumenty prędkość i czas jazdy. Metoda dodaje do kilometrażu przebytą przez samochód drogę wyliczoną na podstawie podanych argumentów. (Dla przypomnienia szkolnych czasów **droga=prędkość*czas** ;))

5. Niektóre metody są wspólne dla każdego obiektu realizowanego w Javie. Jedną z nich jest toString() służąca do wyświetlania informacji o obiekcie na ekranie/W specyfikacji Javy (JAVA API Reference) zapoznać się z opisem metody **toString()** klasy **Object**. Zrealizujemy tu własną metodę **toString() zwracającą** informację o wartościach wszystkich pól obiektu klasy Samochód.

<u>Pierwsze pięć punktów powinny być zrealizowane, aby otrzymać pozytywną ocenę z tych laboratoriów (czyli 3).</u>

Klasa Samochód nie ma metody main, czyli nie możemy jej uruchomić. Możemy jednak tworzyć obiekty danej klasy w innej klasie, mającej metodę main(). Taką klasę będziemy nazywać klasą testową. Daną klasę tworzymy w oddzielnym pliku o nazwie np. **SamochodTest.** Wszystkie operacje związane z obiektem(-ami) klasy Samochod będą wykonywane w klasie testowej.
Aby otrzymać wyższe oceny powinny być zrealizowane pozostałe nie mniej ważne punkty:

1. Mając klasę testową, w metodzie main stwórzmy kilka obiektów klasy Samochód za pomocą konstruktora z argumentami. Obiek klasy tworzy się za pomocą konstrukcji **Klasa obiekt = new Klasa([parametry]);** (zapis [ ] oznacza, że parametry mogą nie być wymagane, czyli może być wywołany pusty konstruktor). Sprawdźmy,jak działa własna realizacja metody **toString()** za pomocą **System.out.println(objekt)**. Spróbujmy również bezpośresnio zmienić wartości pól obiektów (za pomocą konstrukcji **obiekt.nazwaPola=wartość**, a w szczególności pól prywatnych). Co sygnalizuje NetBeans ? Przy pomyślnych zmianach wyświetlmy ponownie pełną informację o obiektach klasy Samochód. Po wykonaniu tego zadania mamy zapewnioną ocenę 4 ☺

2. No i tradycyjnie na zakończenie - tworzenie tablicy z obiektami własne klasy. Tablicę obiektów opisujemy w tym zadaniu tak: **Klasa tablica[] = new Klasa[rozmiar_tablicy]**. Stwórzmu tablicę obiektów klasy Samochód o pojemności np. 10 i zapełnijmy ją obiektami klasy Samochód (obiekty tworzymy za pomocą konstruktora bez argumentów) przy pomocy dowolnej pętli. Za pomocą kolejnej pętli powinniśmy wyświetlić pełną informację o wszystkich obiektach znajdujących się w tablicy. (O wyświetleniu mowa w poprzednim punkcie). Po wykonaniu tego zadania mamy zapewnioną ocenę 5 ☺