Klasy abstrakcyjne i interfejsy

Wyobraźmy sobie klasy Samochód oraz Rower. Obie klasy reprezentują różne pojazdy, jednak możemy wyodrębnić dla nich część wspólną – oba umożliwiają przemieszczanie się i oba posiadają funkcjonalności takie jak jedź czy zatrzymaj się. Co więcej, właściwość tą posiadają także inne pojazdy oraz na przykład zwierzęta, chociaż każde z nich realizuje poruszanie się na swój własny sposób. Metody związane z poruszaniem się możemy więc wyodrębnić w postaci interfejsu.

Interfejsy są swego rodzaju „kontraktem", który mówi co klasa go implementująca może robić, ale nie podaje jak. Za pomocą interfejsu wskazujemy jakie funkcjonalności będzie posiadać implementująca go klasa. Wszystkie metody w interfejsie są domyślnie publiczne i abstrakcyjne, a co za tym idzie nie podajemy ich definicji w interfejsie, natomiast każda klasa implementująca interfejs musi implementować te metody.

Dodatkowe uwagi: W interfejsie możemy także deklarować pola stałe (domyślnie każde pole będzie publiczne, statyczne i stałe - final). Nie można w nim deklarować metod statycznych. Warto zapamiętać, że dziedziczenie wielokrotne w Javie jest niedozwolone, jednak implementowanie wielu interfejsów jest poprawne.

Zadanie 1.

Utwórz

1. Interfejs Moveable zawierający tylko dwie metody abstrakcyjne:
   - void start();
   - void stop();
2. Klasę Rower implementującą interfejs Moveable

```
public class Rower implements Moveable{
    @Override
    public void start() {
        System.out.println("Rower rusza");
    }
    @Override
    public void stop() {
        System.out.println("Rower zatrzymał sie");
    }
}
```

3. Klasę Samochod z polem marka implementującą interfejs Moveable
4. Interfejs Speakable {

z dwoma stałymi statycznymi

int QUIET = 0; // <- publiczne stałe statyczne

int LOUD = 1; // domyślnie public static final

i metodą abstrakcyjną

String getVoice(int voice);

5. Abstrakcyjną klasę Zwierze z polem private String name = "bez imienia"; konstruktorami, metodą zwracającą wartość imienia (getter), nadpisaną metodą toString i abstrakcyjna metodą: public abstract String getTyp(); //metoda abstrakcyjna- ogolne zwierze nie wiadomo jaki typ.
6. Klasę Pies dziedziczącą po klasie Zwierze i implementującą oba interfejsy.

public String getTyp() {return "Pies"; }

7. Następnie napisz metodę wyscig(...), która jako parametry ma dostać dowolną liczbę obiektów Moveable (skorzystaj z "varargs"), na których powinna uruchomić metodę start().