*źródła: https://www.samouczekprogramisty.pl/interfejsy-w-jezyku-java/*

**Interfejs**

Zestaw metod bez ich implementacji (bez kodu definiującego zachowanie metody)1. Właściwa implementacja metod danego interfejsu znajduje się w klasie implementującej dany interfejs.

Wszystkie metody zawarte w interfejsie domyślnie są publiczne więc w tym przypadku można ominąć słowo kluczowe public, nie jest potrzebne.

**Poza zwykłymi metodami w interfejsie mogą się znajdować**

* metody domyślne - mogą mieć właściwą implementacje w ciele interfejsu. Metody takie poprzedzone są słowem kluczowym default
* metody prywatne - poprzedzone są słowem kluczowym private3. Metody prywatne, w odróżnieniu od pozostałych, mogą być wywołane wyłącznie w definicji interfejsu
* metody statyczne,
* stałe.

Jedna klasa może implementować wiele interfejsów. W takim przypadku klasa implementująca musi definiować metody wszystkich interfejsów, które implementuje.

Interfejsy mogą dziedziczyć po innych interfejsach. Interfejs, który dziedziczy po innych interfejsach zawiera wszystkie metody z tych interfejsów.

**Interfejs znacznikowy** - służy on do oznaczenia, danej klasy. Dzięki temu możesz przekazać zestaw dodatkowych informacji. Przykładem takiego interfejsu jest java.io.Serializable, którego używamy aby dać znać kompilatorowi, że dana klasa jest serializowalna (o serializacji przeczytasz w innym artykule).

Każdy obiekt w języku Java może być przypisany do zmiennej określonego typu, jest to jego klasa. Interfejsy pozwalają na przypisane obiektu do zmiennej typu interfejsu.

Interfejsy są idealnymi **mixinami**. Upraszczając, mixin jest typem, który klasa może (dodatkowo do swojego podstawowego typu) zaimplementować, dostarczając dodatkową, odseparowaną funkcjonalność. Mixin sam w sobie nie może zostać zinstancjonowany - jest tylko częścią funkcjonalności innych klas. Dla przykładu, Comparable jest mixinem, który deklaruje i dodaje klasie funkcjonalność porównywania i tym samym porządkowania jej instancji. Klasy abstrakcyjne nie mogą być mixinami z tego samego powodu co wcześniej - klasa może rozszerzać tylko jedną klasę.

**Klasa abstrakcyjna vs interfejs:**

- klasa może rozszerzać tylko jedną klasę abstrakcyjną, a implementować wiele interfejsów

-