Table of Contents

[1. Java interfaces 2](#_Toc4260263)

[2. Exemplu 4](#_Toc4260264)

# Java interfaces

1. O interfață se realizează prin utilizarea “interface”
2. Nu se poate instanția
3. Oferă abstractizarea maximă, nu conține corp la metode ( cel puțin până la 1.7, inclusiv)
   1. Metode default
   2. Metode statice
4. Nu au constructor
5. Orice atribut al unei interfețe este implicit public static final
6. Orice interfață este implicit public și abstract
7. Interfața nu poate extinde o clasă, însă poate extinde o alta interfață
8. O clasă pentru a implementa o interfață utilizează cuvântul cheie implements
9. O clasă ce implementează o interfață trebuie neapărat să implementeze toate metodele sau să fie abstractă

*Avantaje*

* Oferă un contract pentru implementarea claselor, ce vor fi obligate să îl respecte
* ierarhie de clase se creează ușor plecând de la interfețe
* Pentru că java poate implementa mai multe interfețe, însă o singură clasă, este mai bine să se folosească interfețele ca punct de plecare

*Dezavantaje*

* Metodele ce sunt folosite în interfețe trebuie alese cu grijă altfel orice modificare ulterioară a lor, va duce la modificări în toată ierarhia
* În cazul în care clasele ce implementează o interfață e necesar să se folosească și alte metode, diferite de cele din interfață, este necesar cast la clasa concretă.

Exemplu de interfață

**public interface** Shape {  
  
 String ***intern*** = **"aceasta este o variabila public static final"**;  
  
 **default void** doDefaultMethod() {  
 System.***out***.println(**"Shape -> doDefaultMethod()"**);  
 }  
  
}

*Categorii speciale de interfețe*

* Interfețe marker

ex. Serializable, Clonable

* Interfețe funcționale

ex. Suplier, Consumer

<https://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html>

„Hiding implementation is not just a matter of putting a layer of functions between the variables. Hiding implementation is about abstractions! A class does not simply push its variables out through getters and setters. Rather it exposes abstract interfaces that allow its users to manipulate the essence of the data, without having to know its implementation.”

sursa

Clean Code – Robert C. Martin

Data/Objecy anti-symmetry

„These two examples show the difference between objects and data structures. Objects hide their data behind abstractions and expose functions that operate on that data. Data structure expose their data and have no meaningful functions. Go back and read that again. Notice the complimentary nature of the two definitions. They are virtual opposites. This difference may seem trivial, but it has far-reaching implications.”

sursa

<http://en.wikipedia.org/wiki/Law_of_Demeter>

# Exemplu

Triangle (class) impl Shape

Double sideA

Double sideB

Double sideC

Rectangle(Double sideA, Double sideB, Double sideC)

getArea();

getPerimeter();

NamedObject (interface)

getName();

getDescription();

TriangleScalene (class) ext Triangle

impl NamedObject

- String name

- String description

TriangleScalene (Double sideA, Double sideB, Double sideC)

getName();

getDescription();

Shape (interface)

getArea();

getPerimeter();

Square (class) ext Rectangle

Square (Double side)

Rectangle (class) impl Shape

Double sideA

Double sideB

Rectangle(Double sideA, Double sideB)

getArea();

getPerimeter();