

# INTERFACE VS ABSTRACT CLASS

Type of methods	Interface can have <b>only abstract methods</b>	Abstract class can have <b>abstract and non-abstract methods</b>
Type of variables	Interface has only <b>static and final variables</b> .	Abstract class can have <b>final, non-final, static and non-static variables</b> .
Implementation	Interface <b>can't</b> provide the <b>extension of abstract class</b> .	Abstract class <b>can</b> provide the <b>implementation of interface</b>
Inheritance vs Abstraction	A class <b>implements</b> an interface	A class <b>extends</b> an abstract class
Multiple implementation	Interface can <b>extend</b> another Java <b>interface only</b>	Abstract class can <b>extends another</b> Java <b>class</b> and <b>implements multiple</b> Java <b>interfaces</b> .
Accessibility of Members	<b>Members</b> of an interface are <b>public</b> by default.	Abstract class can have class members like <b>private, protected, etc.</b>
Meaning	Interface represents a <b>behaviour</b> of a class (eg. Comparable, Comparator, Serializable, Cloneable, etc...)	Abstract class represents a (partial) <b>entity</b>
Commons	Interfaces <b>cannot be instantiated</b>	Abstract classes <b>cannot be instantiated</b>

# ESEMPIO SU ASTRAZIONE

Modellizzare in Java il seguente problema.

## PARTE 1

Si vuole realizzare un sistema di gestione di un magazzino in cui vengono stoccate solo scarpe da ginnastica.

Ogni scarpa è caratterizzata da:

- Codice
- Marca
- Modello
- Data di stoccaggio
- Prezzo
- Misura
- Set di colori

Inoltre è necessario conoscere il volume di ingombro di ogni scatola di scarpe e le dimensioni spaziali di ingombro (altezza, larghezza e profondità) del contenitore.

Il codice di ogni paio di scarpe è univoco.

In ogni momento il magazzino deve poter:

- A. Fornire la propria capienza massima
- B. Stabilire quante scarpe sono presenti
- C. Stabilire l'ammontare del volume occupato da tutte le scatole di scarpe
- D. Stabilire il peso di tutte le scatole di scarpe presenti
- E. Aggiungere una nuova scarpa
- F. Rimuovere un paio di scarpe
- G. Dato un codice, restituire la scarpa associata
- H. Fornire la propria locazione geografica

## PARTE 2

Riflettere su: chi ha il compito di generare il codice?

Applicare il principio SOLID di Singola Responsabilità.

## PARTE 3

Aggiungere la possibilità di immagazzinare ogni genere di merce (si supponga sempre confezionata in scatole). Per ogni tipo di merce si deve poter conoscere:

- Il codice
- Le dimensioni di ingombro
- La data di stoccaggio
- Il peso
- La tipologia (solido, liquido, gas)
- La misura (Kg, litri, m<sup>3</sup>,...in base al tipo di merce)

Tutte le operazioni della PARTE 2 devono essere riviste prevedendo qualunque tipo di merce.

Modificare il punto C in questo modo:

C. Stabilire l'ammontare del volume occupato da tutte le merci e gli scaffali di supporto

## PARTE 4

Aggiungere la possibilità di inviare tramite una POST al servizio

<http://store.my.warehouse.com> le principali informazioni relative al magazzino:

- Locazione geografica
- Percentuale di riempimento (scaffali compresi)
- Numero complessivo di merci presenti