**Interfete**

*Interfata* = tip de data de tip referinta prin care se poate modela comportamentul unei clase.

*Declarare :*

[public] interface NumeInterfata{  
 constante *(date membre public + static + final* ***– implicit****)*

<= 1.7

metode abstracte *(public + abstract* ***– implicit****)*

metode implicite *(default)* *( sunt metode cu implementare completa + specificatorul public)*

<= 1.8

metode statice *(public* ***-implicit****)*

<= 1.9

metode private *metode cu implementare + static / nestatic )*

}

*Implementare:*

class NumeClasa implements Interfata\_1, Interfata\_2, ... { .... }

Ex: interfata Comparable -> sortare naturala

class Persoana implements Comparable <Persoana> {

int compareTo(Persoana p){ *// metoda abstracta din interfata Comparable*

return this.nume.compareTo(p.nume); *// compareTo-ul asta e din clasa String*

Persoana[] p = new .. ;

! O interfata **nu** poate fi instantiata ! ex:

class **Angajat** extends **Persoana** implements **ContBancar** { ... }

1. Angajat a = new Angajat(...); *// are acces maxim.*
2. Persoana a = new Angajat(...); *// comp. Polim. Nu are acces la angajat*
3. ContBancar a = new Angajat (...); *// am acces doar la metodele implementate din interfata + la metodele din clasa Object*

Arrays.sort(p); *// v-a folosi compareTo pt a sorta*

*Implementarea intr-o clasa a mai multor interfete:*

*InterfataA |InterfataB |Clasa*

Int suma (int x, |int suma (int x, |int suma(x,y)

int y) | int y, int z) |int suma(x,y,z)

int suma(x,y) |int suma(x,y) |int suma(x,y)

int suma(int x, |double suma |nu se poate

int y) | (int x,in y) | implementa

! Interfetele pot fi extinse !

Cum se poate simula mostenirea multipla : prin compozitie

Class A

Class B 🡪b .. *compozitie*

Class C 🡪c .. *compozitie*

*Void afisare(){*

*b.afisare(); // se poate numi “ mostenire “*