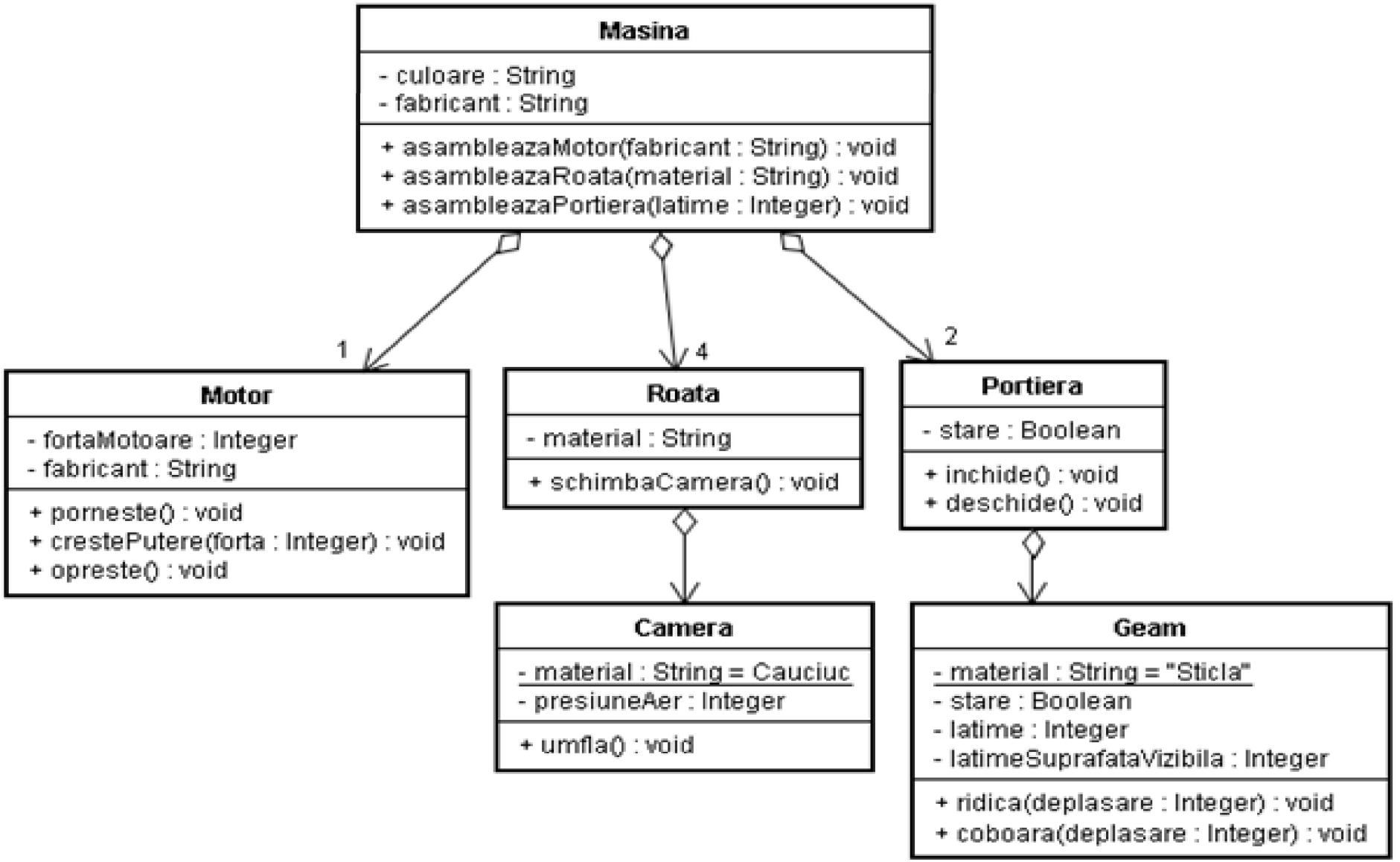
Curs 7 POO

© Conf. univ dr. Crenguta M. Puchianu

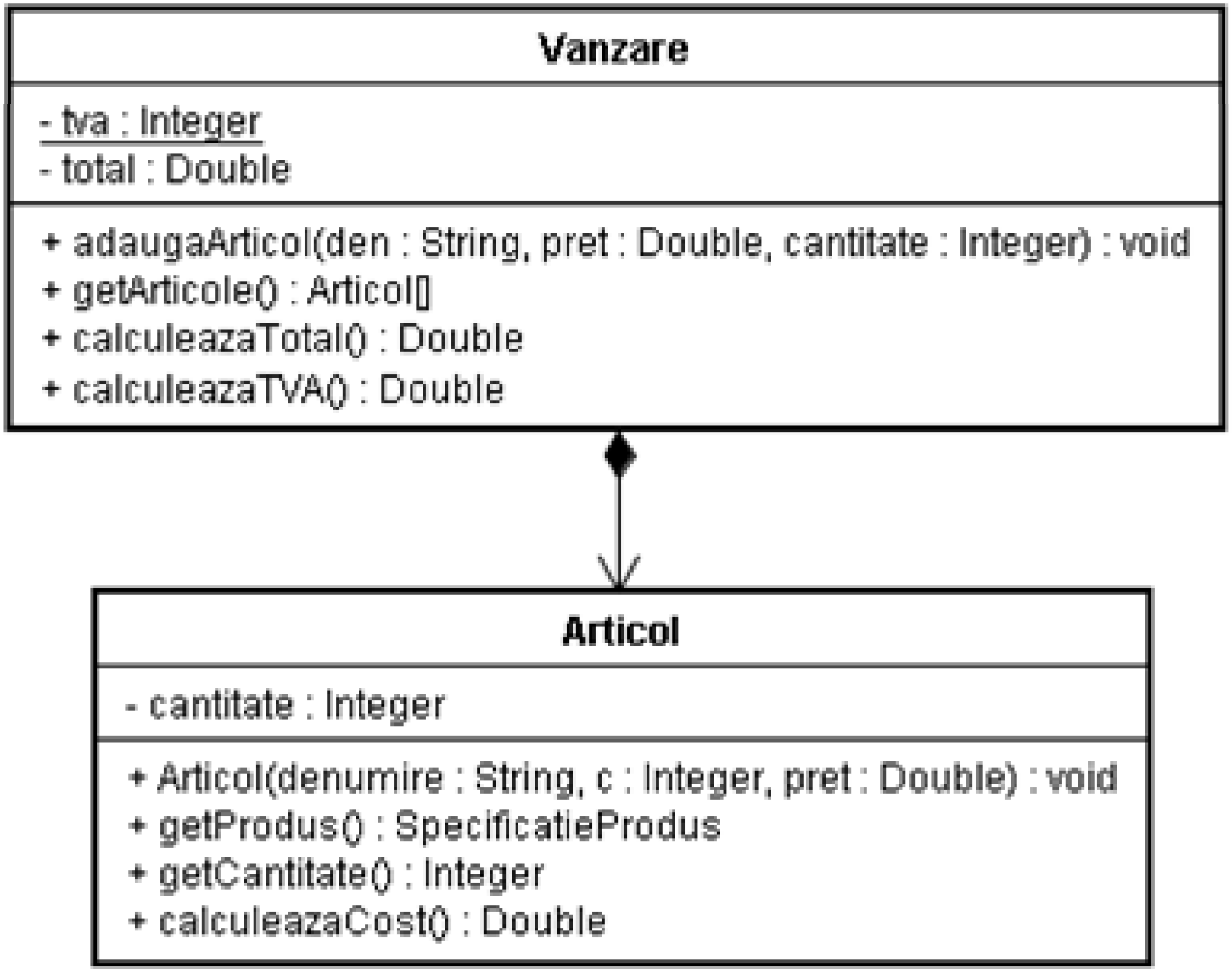
- Relatia de agregare
- Relatia UML de compozitie
- Relatia UML de generalizare
- Mecanismul de mostenire
- Clasa Object

2023

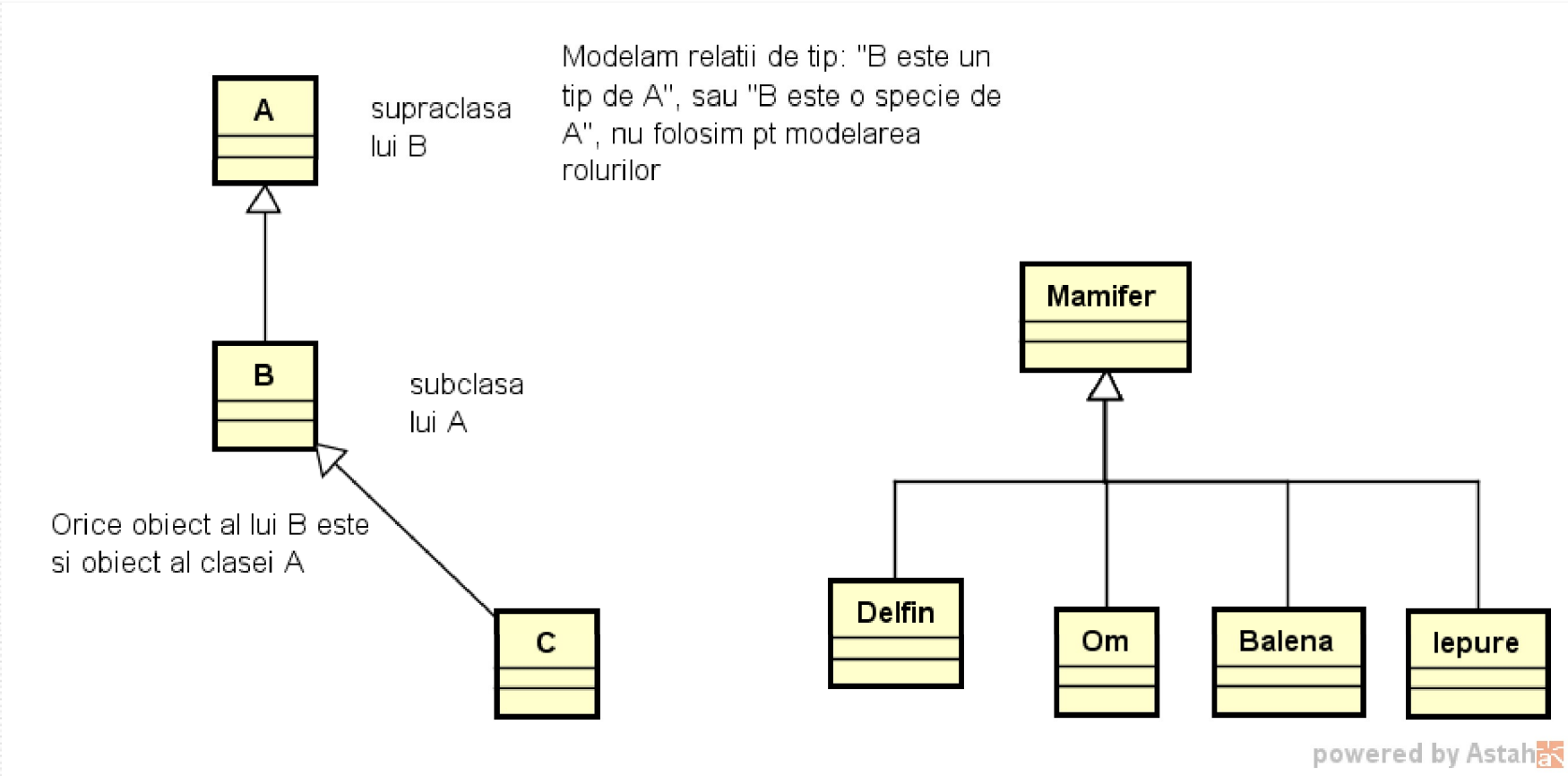
Relatia de agregare



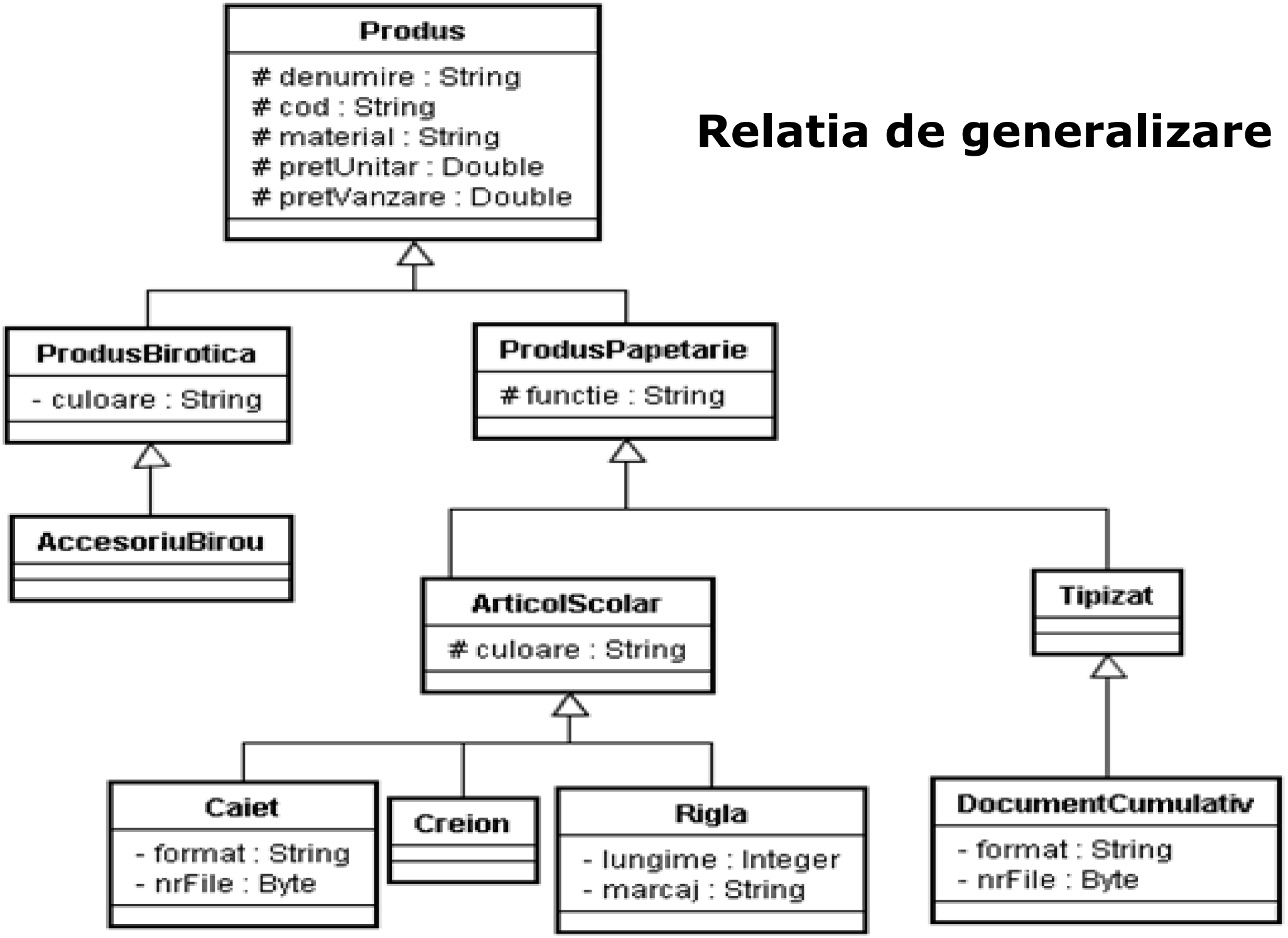
Relatia de compozitie



Relatia UML de generalizare



Relatia de generalizare



Mecanismul de mostenire

- Pentru a reprezenta relaţia de generalizare în limbajele de programare spre obiecte, precum Java, se propune un mecanism numit **moştenire**. Acest mecanism permite unei noi clase să beneficieze de structura şi comportamentul definite într-o clasă deja existentă prin declararea că moştenim acea clasă.

- În Java, mecanismul de moştenire se numeşte **extensia** clasei existente şi cere subclasei să declare ce clasă extinde. Apoi, subclasa respectivă trebuie să definească propriile variabile şi metode.

- Declararea că o clasă extinde o altă clasă este indicată în Java prin cuvântul cheie extends în declaraţia clasei respective:

```
[modificator] class NumeSubClasa extends NumeSupraClasa {
//structura si comportamentul proprii
}
```

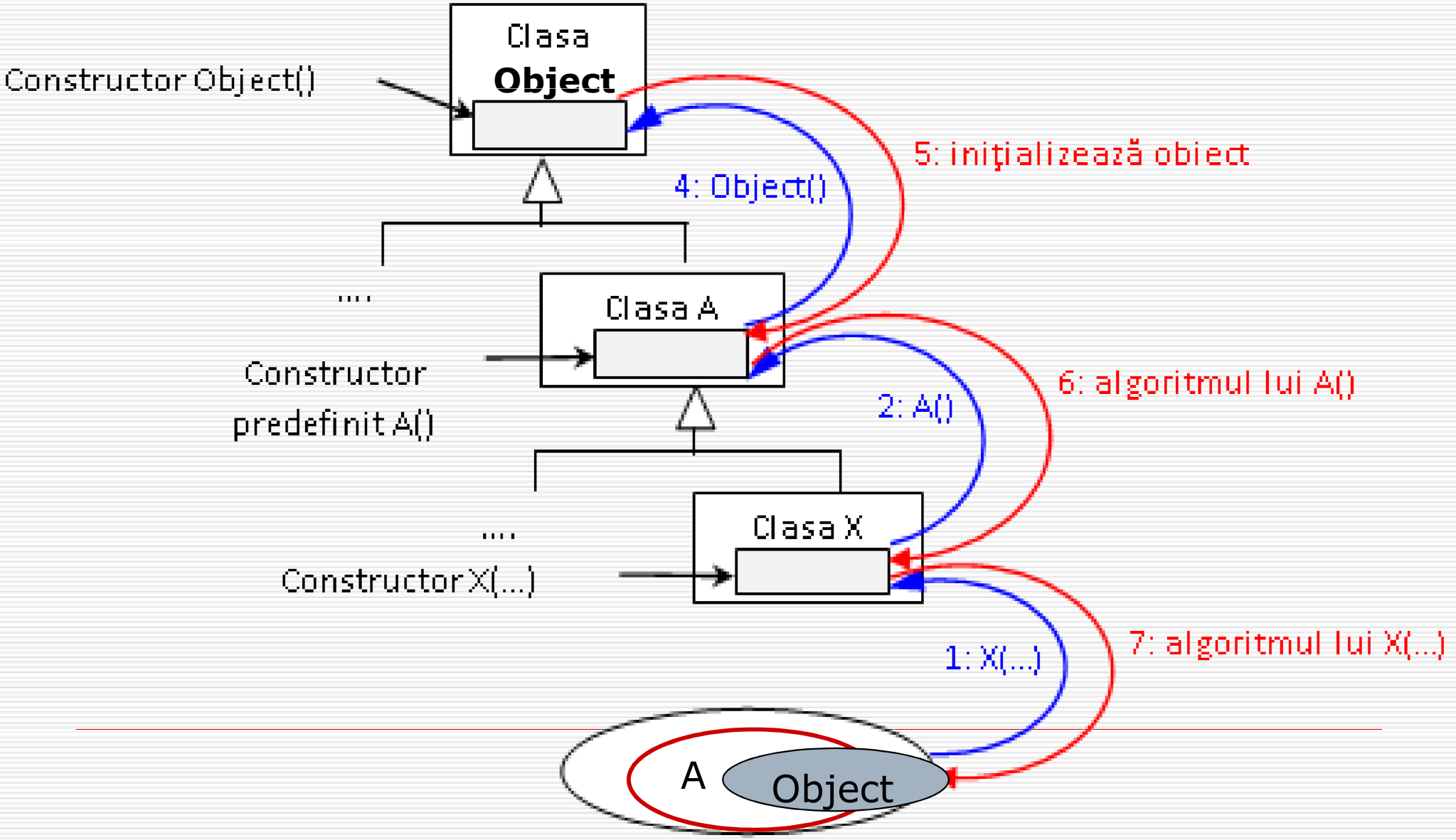
Regulă. În Java, o clasă poate extinde o singură clasă. Aşadar, moştenirea este simplă.

Ierarhia claselor Java

- În Java, clasele formează o structură de arbore în care rădăcina este clasa Object, a cărei definiţie apare în pachetul java.lang. Orice clasă extinde direct (implicit sau explicit) sau indirect clasa **Object**. Aşadar, toate clasele definite de programator care nu extind o altă clasă sunt automat subclase ale clasei Object.

```
public class Object {
  public java.lang.Object();
  public java.lang.String toString();
  protected  native java.lang.Object clone() throws
CloneNotSupportedException;
  public boolean equals(java.lang.Object o);
  public int hashCode()
  protected void finalize() throws Throwable;
  public final java.lang.Class getClass();
  public final native void notify();
  public final native void notifyAll();
  public final void wait() throws InterruptedException;
  public final native void wait(long) throws InterruptedException;
}
```

Mecanismul de initializare a obiectelor



Utilizarea obiectelor subclaselor

- Regula: Daca intr-o variabila de tip supraclasa unei clase este memorata referinta unui obiect al subclasei respective, atunci cu variabila respectiva putem numai metodele declarate/definite in supraclasa clasei respective.

```
ProdusPapetarie p2=new ProdusPapetarie("pic", "1234", "plastic",
  0.75, "de sters");
System.out.println(p2.calculeazaPretVanzare()+" "+
                                                p2.getFunctie());
Produs p3=p2;//p3 refera un obiect al ProdusPapetarie
System.out.println(p3.calculeazaPretVanzare());
//la compilare se verifica daca metoda calculeazaPretVanzare a
  fost declarata in Produs. La executie, se executa metoda
calculeazaPretVanzare mostenita de catre obiectul referit de p3
```

- Principiul substituirii: Orice obiect al subclasei poate inlocui un obiect al supraclasei sale fara sa genereze mesaje de eroare la compilare.
- Observatie. Daca o clasa este declarata cu modificatorul **final**, atunci ea nu poate fi extinsa de o alta clasa.