PROGRAMARE ORIENTATĂ PE OBIECTE

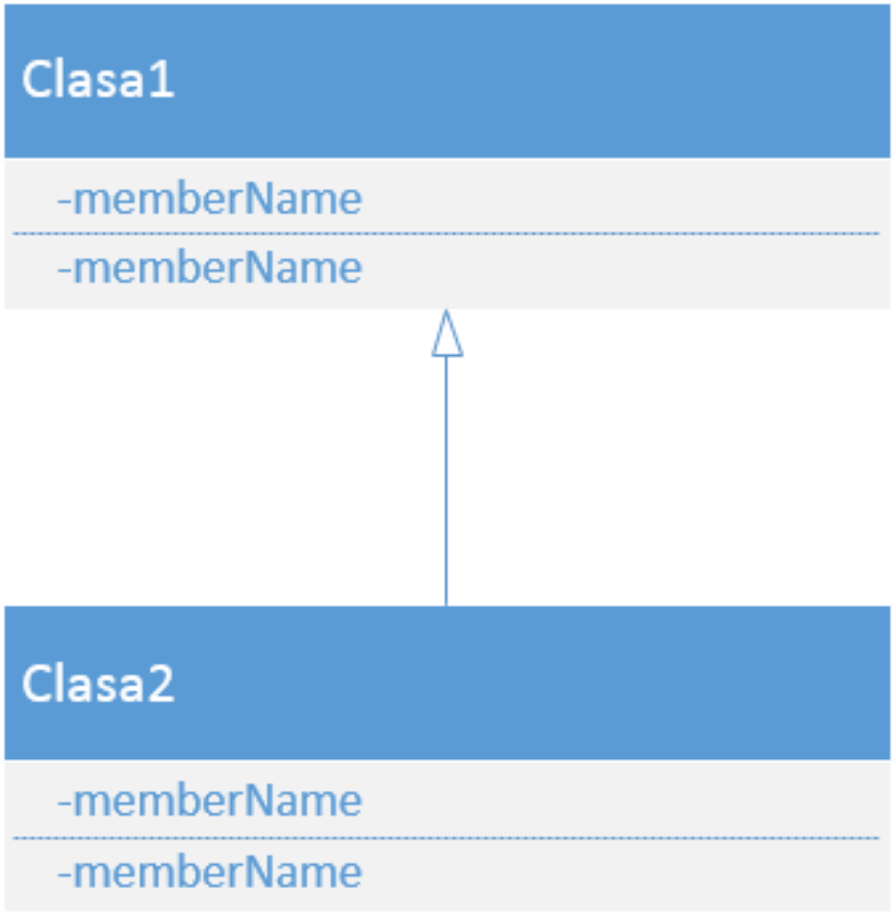
GENGE BÉLA

Capitolul 5
Moștenirea. Supraîncărcarea și supradefinirea metodelor.

Moștenirea

- Moștenirea este o proprietate fundamentală în POO.
- Reprezintă mecanismul de specializare (extindere, personalizare) a unei clase.
- Terminologie:
  - Clasa din care se realizează moștenirea: clasa de bază, clasa părinte, super-clasa.
  - Clasa care realizează moștenirea: clasa derivată.

Moștenirea: reprezentare UML



Moștenirea

- Moștenirea asigură extinderea proprietăților unei clase (câmpuri și metode).

- Moștenirea permite înlocuirea metodelor clasei de bază (suprascriere – supradefinire).
  - Se poate realiza doar pentru metode **virtuale**.
  - **În Java toate metodele sunt virtuale**.

Exemple de reprezentări UML

- FormaGeometrica - Patrat.
- IAutomobil – Renault – Dacia – Logan.
- Structuri de date (lista, stiva, coada).

Implementare în Java

- Java NU suportă moșternirea multiplă!
  - O clasă poate moșteni o singură super-clasă. Atenție! Este posibilă totuși crearea unei ierarhii mai complexe.
- Moștenirea se realizează prin cuvântul cheie *extends* .
- Cuvântul cheie *super:*
  - Asigură accesul asupra proprietăților (câmpuri, metode, constructor) clasei de bază.

Implementare în Java – reguli

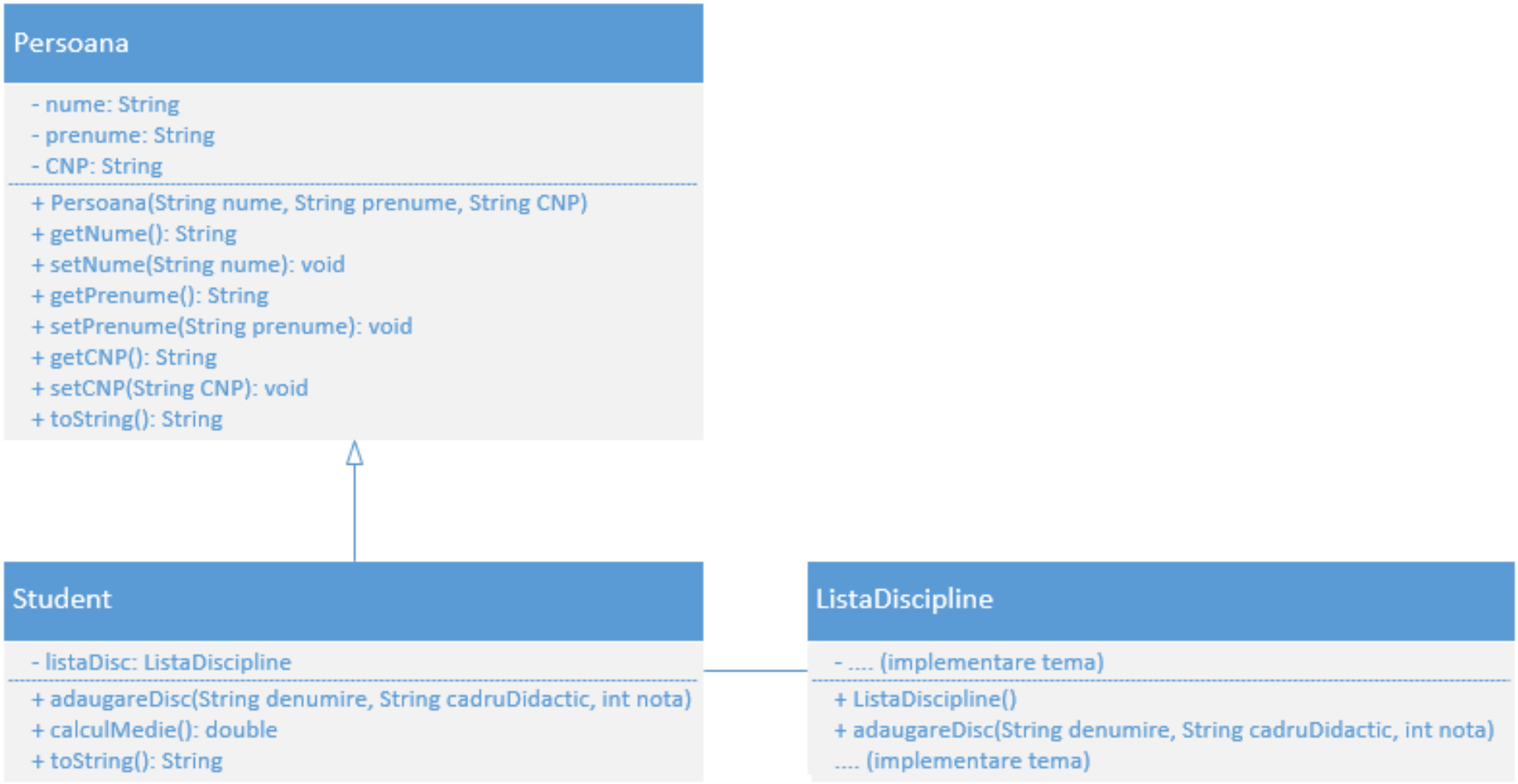
- Dacă în clasa derivată definim un constructor, în cadrul acestuia pe **prima linie** se va apela în mod explicit constructorul clasei de bază:
  - super(parametrii_constructor_baza).
- În mod obligatoriu, clasa derivată va implementa (supradefini) metoda **toString()**.

Controlul accesului

	Clasă	Clasă în același pachet	Sub-clasă în alte pachete	Orice clasă
public	**DA**	**DA**	**DA**	**DA**
protected	**DA**	**DA**	**DA**	nu
implicit	**DA**	**DA**	nu	nu
private	**DA**	nu	nu	nu

Exercițiu

- Să se implementeze diagrama UML:



Instanțierea

- Student st1 = new Student(...);
- Persoana st2 = new Student(...);
- Persoana st3 = new Persoana(...);
- Student st4 = new Persoana(...);

- st1. toString();
- st2. toString();
- st3. toString();
- st4. toString();

Metode și clase finale

- Metodele finale (prin cuvântul cheie *final* înaintea tipului returnat) NU mai pot fi suprascrise.
  - Exemplu: public final adaugareDisc(...).

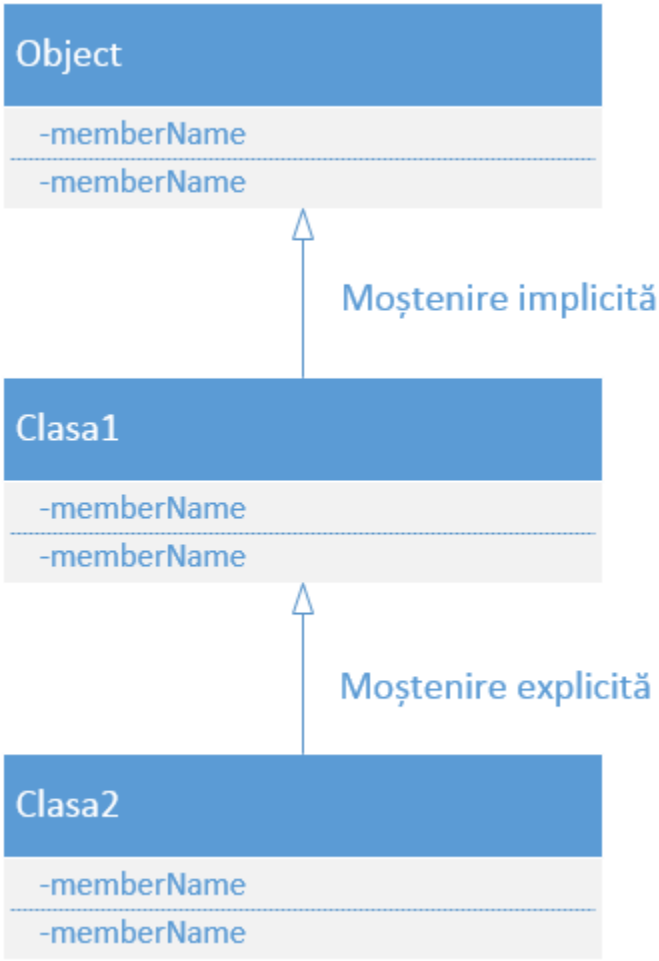
- Clasele finale (prin cuvântul cheie final plasat înaintea declarării clasei) NU mai pot fi derivate.
  - Exemplu: final class Student {…}

Supraîncărcarea

- Reprezintă proprietatea POO prin care aceeași metodă este definită de mai multe ori.
- Diferențierea metodelor supraîncărcate se realizează prin numărul și/sau tipurile de parametrii.
- Exemplu: metoda **print** din clasa PrintStream (accesat prin câmpul static **out** din clasa System):
  - void print(char ch);
  - void print(String s);
  - void print(double d):
  - void print(int i);

Clasa Object

- Este clasa din care sunt derivate în mod implicit toate clasele.
- În Java o clasă poate deriva o singură altă clasă, dar implicit în ierarhia de moștenire, prima clasă va deriva clasa Object (implicit).



Clasa Object

- Metode ce pot fi suprascrise:
  - wait()
  - notify()
  - notifyall()
  - getClass()
- Se recomandă suprascrierea metodelor:
  - String toString()
  - boolean equals(Object o)
- Exemplu de implementare a metodelor toString() și equals() pentru clasa Persoana.

Exemplu extins (care va fi rezultatul in fiecare caz?):

Student s1 = new Student(…); Student s2 = new Student(…); s1.equals(s2);

sau:

Object s1 = new Student(…); Object s2 = new Student(…); s1.equals(s2);

Clase și metode abstracte

- O clasă se numește abstractă dacă conține cel puțin o metodă abstractă.
- O metodă pentru care nu se specifică o implementare se numește **abstractă**.
  - Se declară prin cuvântul cheie **abstract**.
- O metodă pe care nu o definim la un anumit nivel în ierarhia de moștenire trebuie declarată abstractă.

Clase și metode abstracte

- Asemănarea cu interfețele POO:
  - Interfața include doar metode abstracte.
- Diferența față de interfețele POO:
  - Clasele abstracte pot conține câmpuri care nu sunt statice și finale.
  - Clasele abstracle pot conține câmpuri și metode care nu sunt publice.
  - Clasele abstracte pot conține definiții de metode (metode care nu sunt abstracte).
- Aplicabilitate:
  - Definirea unor clase incomplete într-o ierarhie de clase.
  - În cazul în care se identifică porțiuni/metode de cod comune pentru o suită de clase dar și implementări ale metodelor care sunt specifice pentru anumite situații, metodele comune vor fi definite într-o clasă abstractă.

Clase și metode abstracte

- Exemple:
  - FormaGeometrica.
  - Automobil.