INGINERIA PROGRAMELOR

LUCRAREA DE LABORATOR NR. 5

UTILIZAREA LIMBAJULUI DE MODELARE *UML*

-Diagrama de clase-

Modelarea conceptuală. Diagrama de clase

Modelarea conceptuală (numită şi modelarea domeniului) este activitatea de identificare a conceptelor importante pentru sistem. În tehnica de programare orientată obiect, modelarea conceptuală se realizează prin diagrama claselor, întrucât clasele reprezintă concepte. Diagrama claselor furnizează structura codului care va fi scris. Problema principală este identificarea conceptelor.

Diagramele de clase pot conţine:

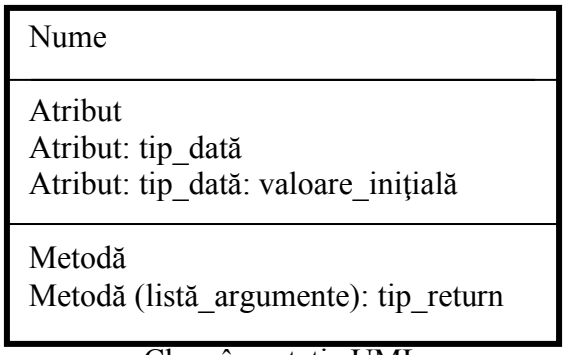
- Clase/Interfeţe
- Obiecte
- Relaţii:
  - Între instanţe ale claselor:
    - Asociere
    - Agregare
    - Compunere
  - Între clase:
    - Moştenire
    - Dependenţă
    - Realizare

Clase, interfeţe, clase parametrizate, obiecte

O clasă modelează vocabularul: identifică conceptele pe care clientul sau programatorul le foloseşte pentru a descrie soluţia problemei.

Pot fi:

- Abstracţii care fac parte din domeniul problemei;
- Clase necesare la momentul implementării;



Clasa în notaţie UML

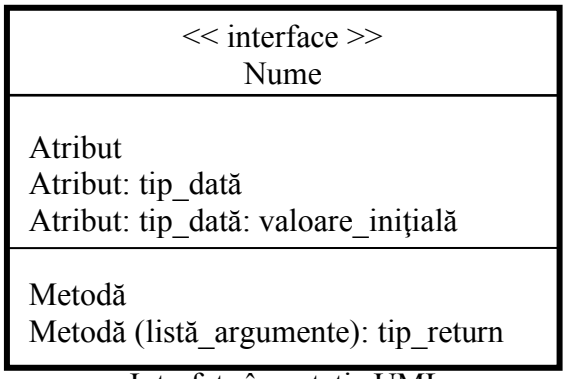
Elementele unei clase sunt:

- Nume: identifică clasa;
- Atribute: proprietăţi ale clasei;
- Metode: implementarea unui serviciu care poate fi cerut oricărei instanţe a clasei.

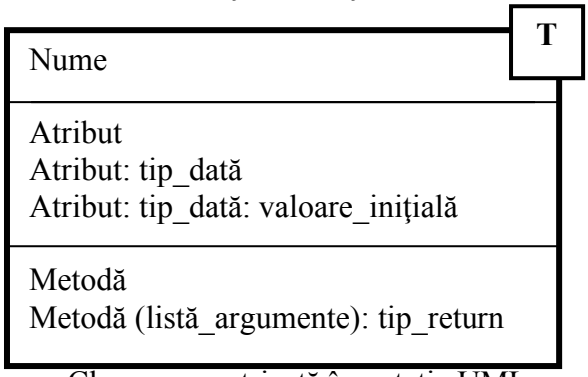
O clasă se reprezintă printr-o căsuţă împărţită în trei. În partea de sus este notat numele clasei, în partea mediană atributele, iar în partea de jos metodele sale. Specificatorii de vizibilitate ai atributelor şi metodelor se reprezintă astfel: + public, - private şi # protected.

Interfaţa specifică o mulţime de operaţii, dar nu menţionează structura internă sau implementarea acestora. O interfaţă poate fi implementată de mai multe clase. O interfaţă se reprezintă printr-o căsuţă împărţită în trei. În partea de sus este precizat cuvântul rezervat *interface* urmat de numele interfeţei, în partea mediană atributele, iar în partea de jos metodele sale.

Clasa parametrizată (template) este o clasă care are unul sau mai mulţi parametri formali, definind o familie de clase. De obicei parametrii reprezintă tipuri ale atributelor.

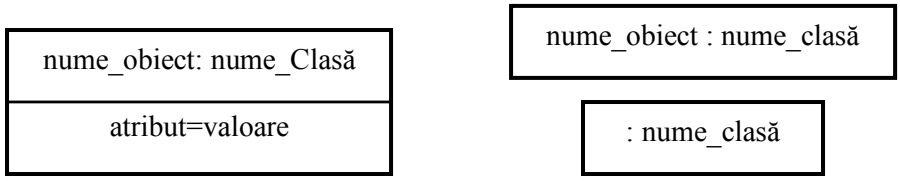


Interfaţa în notaţie UML



Clasa parametrizată în notaţie UML

Un obiect reprezintă o instanţă a unei clase, are identitate şi valori ale atributelor. Un obiect are mai multe reprezentări în notaţie UML prezentate în figura 2.22.

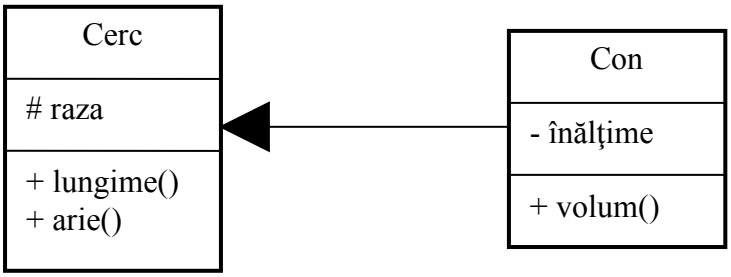


Obiect în notaţie UML

Relaţii între clase

- **Moştenirea**

De multe ori, mai multe clase din arhitectură au atribute şi operaţii comune [16]. Acestea pot fi introduse într-o singură clasă şi moştenite în celelalte. Notaţia UML pentru moştenire este următoarea:



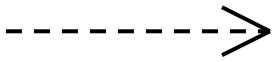
Moştenirea în notaţie UML

O greşeală frecventă în proiectarea orientată obiect este utilizarea abuzivă a moştenirii, ceea ce conduce la probleme în întreţinerea programului. Dacă mai multe clase sunt legate de una singură, schimbările în clasa de bază vor afecta şi clasele derivate. De asemenea, când derivăm o clasă trebuie să parcurgem întreaga ierarhie pentru a vedea ce face implicit clasa respectivă. Această problemă este cunoscută sub denumirea de proliferarea claselor.

Moştenirea nu trebuie folosită decât ca mecanism de generalizare, adică se foloseşte numai dacă clasele derivate sunt specializări ale clasei de bază. De asemenea, toate definiţiile clasei de bază trebuie să se aplice tuturor claselor derivate.

❖ **Dependenţă**

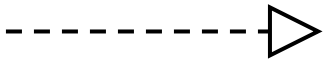
Relaţia de dependenţă modelează interdependenţele ce apar la implementare, indicând o relaţie semantică între două elemente ale modelului. Se foloseşte când o schimbare în elementul destinaţie poate atrage după sine o schimbare în elementul sursă. Reprezentarea acestei relaţii în notaţie UML este prezentată în figura 2.24.



Relaţia de dependenţă în notaţie UML

❖ **Realizare**

Relaţia de realizare se foloseşte când o clasă implementează (realizează) o interfaţă. Reprezentarea acestei relaţii în notaţie UML este prezentată în figura 2.25.



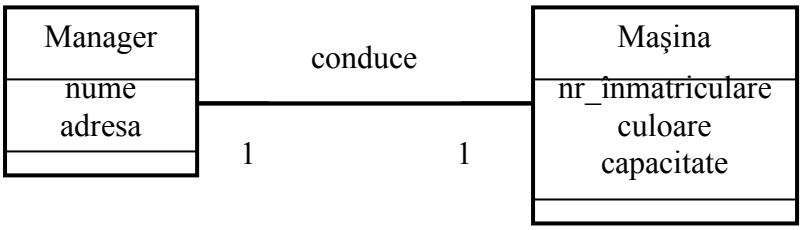
Relaţia de realizare în notaţie UML

Relaţii între instanţe ale claselor

❖ **Asocierea**

Asocierea exprimă o conexiune semantică sau o interacţiune între obiecte aparţinând diferitelor clase [18]. Pe măsură ce sistemul evoluează noi legături între obiecte pot fi create sau legături existente pot fi distruse.

Linia simplă în UML are rolul de asociere. Numerele descriu cardinalitatea asocierii, adică ne spun câte instanţe sunt permise din fiecare concept. Următoarea figură prezintă câteva cardinalităţi posibile, deşi din punct de vedere al notaţiei nu există restricţii asupra cardinalităţilor care pot fi specificate.



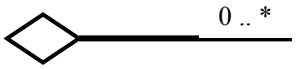
Asocierea în notaţie UML

O greşeală care poate fi făcută în această fază este să decidem că există o relaţie între două concepte, să trasăm o linie între ele, dar să nu notăm tipul de asociere. După ce vom trasa toate liniile nu vom mai şti ce înseamnă fiecare şi va trebui să o luăm de la început.

❖ **Agregarea**

Un aspect important al proiectării orientate obiect este agregarea, ideea că un obiect poate fi construit din altele [18]. De exemplu, un calculator este o agregare între procesor, placă video, placă de sunet etc.

Agregarea este un caz particular al relaţiei de asociere, modelând o relaţie de tip parte-întreg. Această relaţie poate avea toate elementele unei relaţii de asociere, însă în general se specifică numai multiplicitatea.



Agregarea în notaţie UML

❖ **Compunerea**

Compunerea este un concept similar cu agregarea, însă mai puternic deoarece implică faptul că un întreg nu poate exista fără părţi. În exemplul de agregare de mai sus, dacă se înlătură placa de sunet, calculatorul rămâne calculator. Însă o carte nu poate exista fără pagini; o carte este compusă din pagini. Notaţia este asemănătoare, dar rombul este plin.

Compunerea în notaţie UML

II. Aplicaţii prezentate

1. Se doreşte dezvoltarea unui software interactiv destinat studiului arborilor binari. Diagrama de clase restrânsă este prezentată în figura 1, iar diagrama între instanţele claselor este prezentată în figura 2.

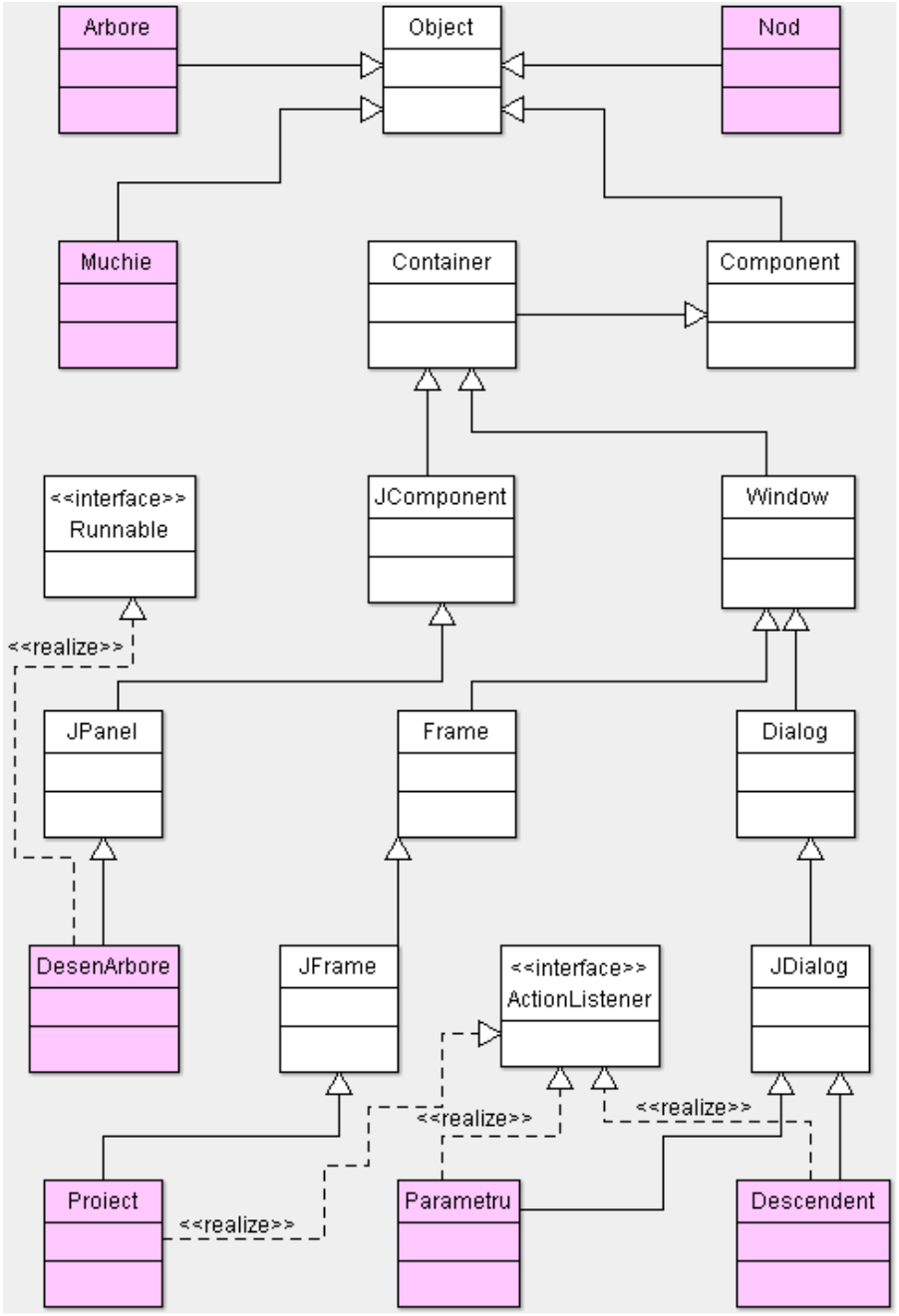


Figura 1 Diagramă de clase

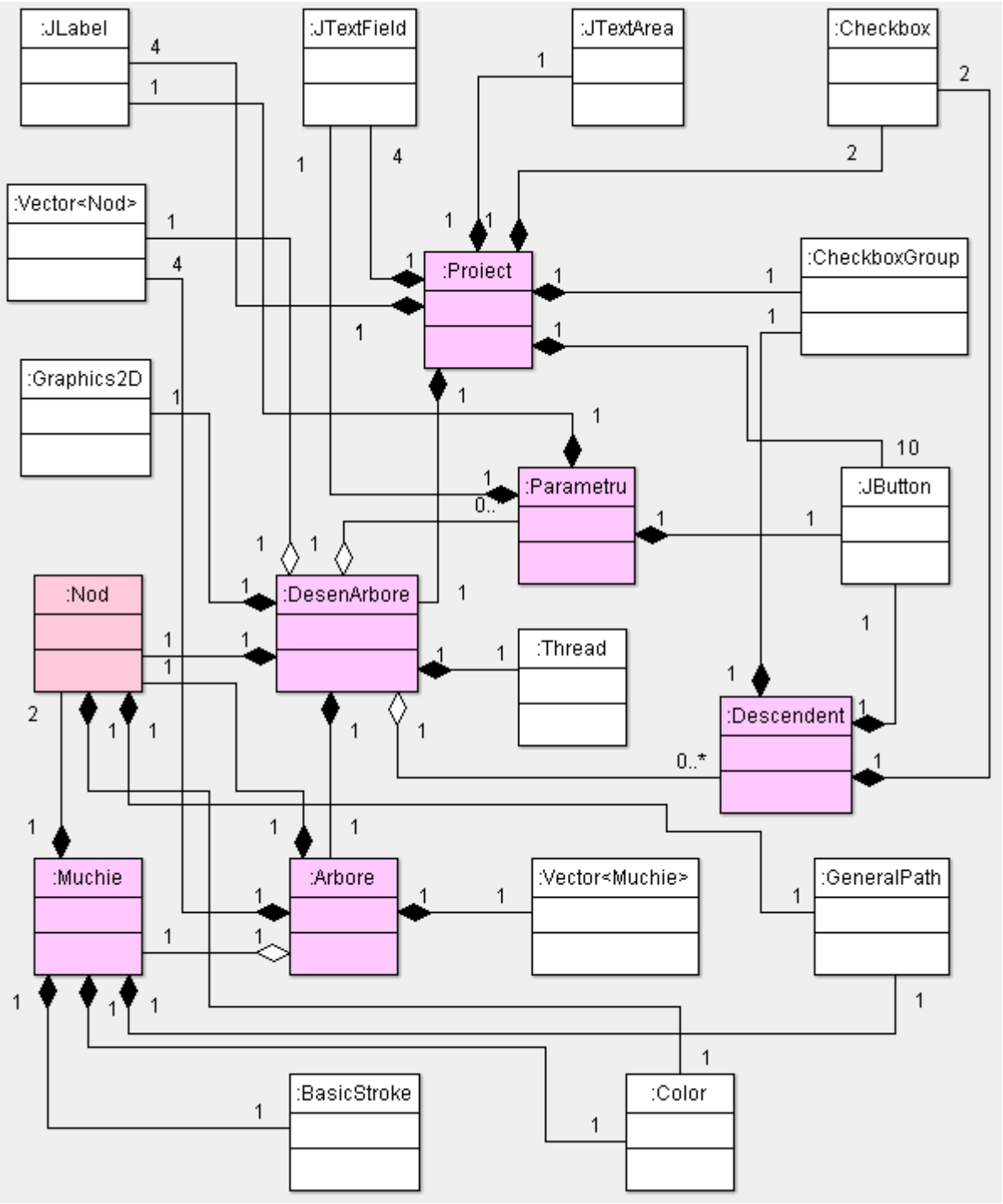


Figura 2. Diagramă între instanţe ale claselor

III. Aplicaţii propuse

Pentru aplicaţiile propuse în laboratorul 1 pentru care s-a realizat diagrama cazurilor de utilizare realizaţi diagramele de clase. Diagramele prezentate anterior, cât şi cele ce realizate pentru aplicaţiile propuse, se vor reprezenta utilizând instrumentul **ArgoUML** sau **Draw.io**