2011 - 2012

Programare **O**rientata **spre O**biecte
(*Object-Oriented Programming*)

a.k.a. Programare Obiect-Orientata

Titular curs: Eduard-Cristian Popovici

Suport curs: *http://discipline.elcom.pub.ro/POO-Java/*

Structura cursului

Continut curs Programare Orientata spre Obiecte (in Java)

1. Introducere in abordarea orientata spre obiecte (OO)
- **1.1. Obiectul cursului si relatia cu alte cursuri**
- 1.2. Evolutia catre abordarea OO
- 1.3. Caracteristicile si principiile abordarii OO
- 1.4. Scurta recapitulare a programarii procedurale/structurate (introducere in limbajul Java)

2. Orientarea spre obiecte in limbajul Java
- 2.1. Obiecte si clase. Metode (operatii) si campuri (atribute)
- 2.2. Particularitati Java. Clase de biblioteca Java (de uz general)
- 2.3. Clase si relatii intre clase. Asociere, delegare, agregare, compunere
- 2.4. Generalizare, specializare si mostenire
- 2.5. Clase abstracte si interfete Java
- 2.6. Polimorfismul metodelor
- 2.7. Clase pentru interfete grafice (GUI) din biblioteca Java Swing

3. Programarea la nivel socket cu Java (pe platforma Java SE)
- 3.1. Clase pentru fluxuri de intrare-iesire (IO)
- 3.2. Introducere in Protocolul Internet (IP) si stiva de protocoale IP
- 3.3. Socketuri flux (TCP) Java.
- 3.4. Clase Java pentru programe multifilare. Servere TCP multifilare
- 3.5. Socketuri datagrama (UDP) Java

Modul de evaluare

10% **prezenta** la **laborator**

10% **teme de casa** la **laborator**

20% **testari** la **laborator**

20% **mini-proiect**

– predare la examenul final

40% **examen final** (scris)

– grila +

– programe de scris/comentat

\+ **bonusuri**

– activitate laborator

– teme suplimentare laborator

– eseuri pe tematici avansate

– mini-proiecte complexe, etc.

1. **Introducere in abordarea orientata spre obiecte (OO)**

   1.1. Obiectul cursului si relatia cu alte cursuri

Obiectul cursului si relatia cu alte cursuri

Vom discuta

- **Programarea**
  - **Masinile programabile** (suportul programarii) – curs **CID** si **AMP** (sem II)
  - **Programele** de calcul (tinta programarii) – curs **PC** si **SDA** (anul 1)
  - Programarea ca **rezolvare de probleme –** curs **Inginerie *Software*** (?)
- **Orientarea spre Obiecte (OO)**
  - **Evolutia catre OO**
    - Masina programabila si **codul masina** – curs **CID** si **AMP**
    - Limbajele de **asamblare** – curs **AMP**
    - Limbajele de **nivel inalt** (pre-OO) – curs **PC** si **SDA**
    - Tipurile de date abstracte (**ADT**) – curs **SDA**
  - **Orientarea** spre **modelarea realitatii**, entitati bazate pe **responsabilitati** (roluri), **incapsulare duala**, **interfete** (specificare), componente ***black-box***, **servicii**, etc.

1.1. Obiectul cursului si relatia cu alte cursuri

Programarea

- **Masinile programabile** (suportul programarii)
  - curs **CID** si **AMP** (sem II)

– De ce discutam despre **masini**?

– Cu ce **scop** a aparut calculatorul?

– Ce **sarcini** indeplineste calculatorul?

– Cum e **reprezentata** informatia in calculator?

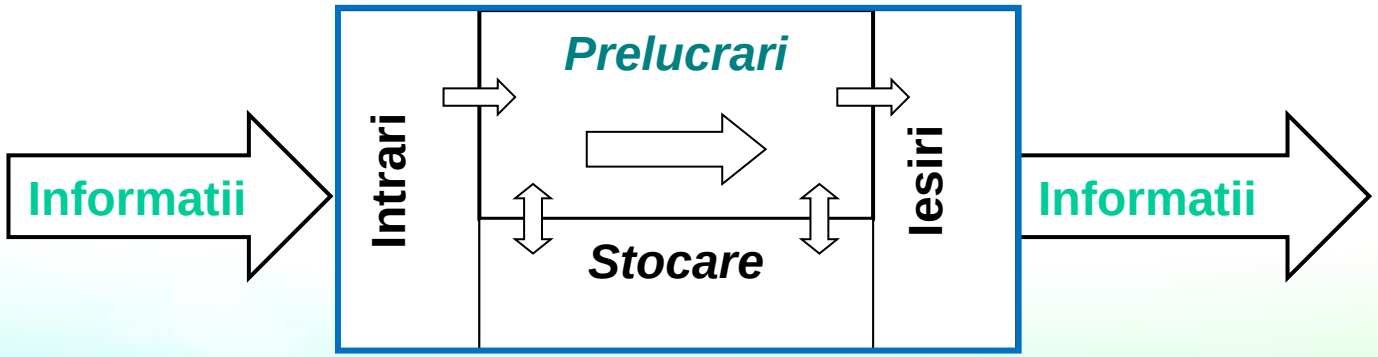
Masinile programabile

Calculatorul

- a aparut cu scopul de a face **calcule matematice** (algoritmice)
- fiind initial o **masina** care ***prelucreaza*** **informatii** (numere, cuvinte, tabele, documente, sunete, imagini, instructiuni, etc.)
  - pe baza unei **secvente de pasi** numita **program** de calcul

    => poate fi considerat o **masina informatica programabila**

Masina programabila

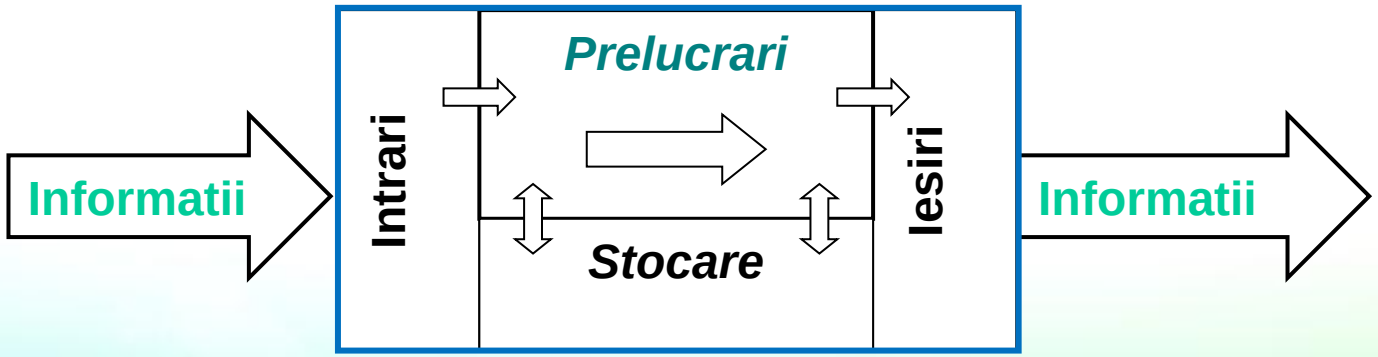


Masinile programabile

***Prelucrarea* informatiilor**

- porneste de la **reprezentarea** acestora
  - **digitala** (secvente de biti – formate din valori 0/1)
- include **stocarea** (memorarea),
- **transferul** (copierea, mutarea) si
- **transformarea** (constructia unor informatii noi din altele vechi)

Masina programabila



Programarea

- **Programele** de calcul (tinta programarii)
  - curs **PC** si **SDA** (anul 1)

– Care sunt **diferentele** intre hardware (HW) si software (SW)?

– Care este **relatia** dintre hardware si software?

– Care este **rolul** software-ului?

– Ce tipuri de **limbaje** exista?

– Ce ar putea insemna **programarea**?

Programele de calcul si masinile programabile

Pentru a oferi servicii *software* utilizatorilor **concureaza**

- **masina** programabila (calculatorul)
  - – sistemul ***hardware*** care ofera **suportul** de calcul si
- **programele** (de calcul)
  - – sistemele ***software*** care **conduc comportamentul** masinii programabile

Programele de calcul

- sunt **secvente** de pasi (numiti **instrucţiuni**) care **conduc acţiunile** unei **maşini** programabile

secventa de pasi (instructiuni)

```
if (x > 1) {
   cout << "impossible" << endl;
} else {
   y = sqrt(1-x);
   cout << "y is " << y << endl;
}
```

Programele de calcul si masinile programabile

Programele de calcul

- sunt **specificatii** ale **comportamentelor viitoare** ale unui **calculator**
  - descriu **raspunsurile** la **cereri** / **evenimente** venite din **exterior** / de la **utilizator**
  - sub forma de **instructiuni** intr-un "**limbaj**" pe care calculatorul il poate intelege (**cod masina** executabil sau **cod de octeti** interpretabil)
    - obtinute eventual prin **conversie** din instructiuni scrise intr-un **limbaj** pe care il poate intelege mai bine **programatorul**

Limbajul de programare

– este un **limbaj** (sistem de conventii adoptate pentru realizarea unei comunicari) **artificial**

– creat pentru a **intermedia comunicatia** intre **programator** si **masina programabila**

– si pentru a **exprima actiuni** efectuate de masina programabila

Programele de calcul si masinile programabile

Modalitatile prin care programatorul poate specifica actiunile unei masini sunt

- **pseudocodul**
  - **descriere compacta** si **informala** de **nivel inalt** a **actiunilor** masinii
  - destinat citirii de catre oameni, nu de catre calculator
    - fiind **usor de inteles** chiar si de catre **ne-programatori**
- **reprezentari vizuale / grafice**
  - **diagrame** (diagrama de flux de date, diagrama MSC, etc.)
  - limbaje de **modelare** vizuala (**UML** – limbajul de modelare unificat)
- **limbajele de programare** clasice, textuale
  - la nivel de **asamblare**
  - de **nivel inalt** (Pascal, C, etc.)

Programele de calcul si masinile programabile

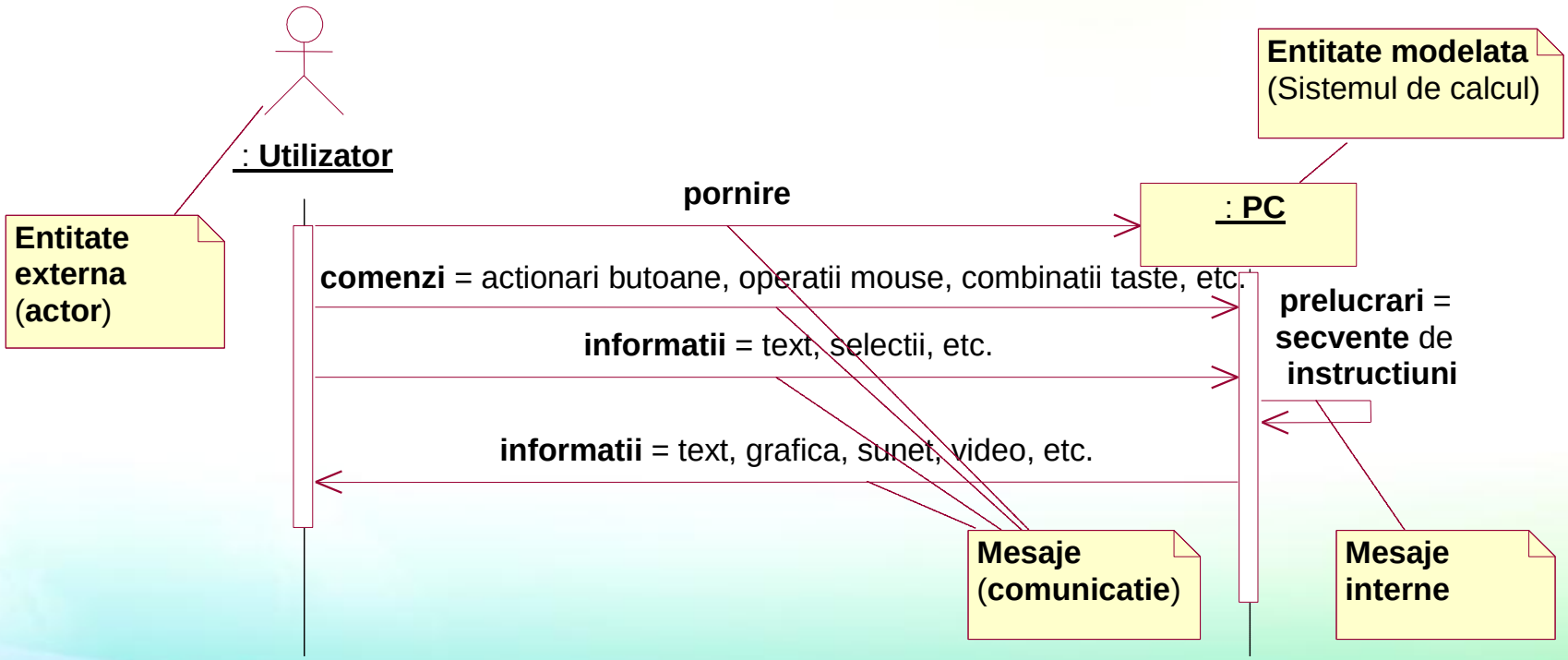
Exemplu de specificatie in “**pseudocod**” a **actiunilor** unei **masini programabile**

- Programul este **pornit**
- **[Repetat pana la indeplinirea unei conditii bazata pe o informatie anume]**
  - Utilizatorul **da comenzi** (actionari butoane, operatii mouse, combinatii taste, etc)
  - Utilizatorul **introduce** (editeaza) **informatii** (prin consola standard de intrare, interfata grafica, etc.)
  - **[Conditionat de valoarea unei informatii anume]**
    - Programul **realizeaza** o secventa de actiuni (**prelucrari**) asupra informatiilor detinute
  - Programul **afiseaza informatii** pe ecran (prin consola standard de iesire, interfata grafica, etc.)
  - Programul **primeste** un **mesaj** prin retea
  - …

Programele de calcul si masinile programabile

Programele de calcul

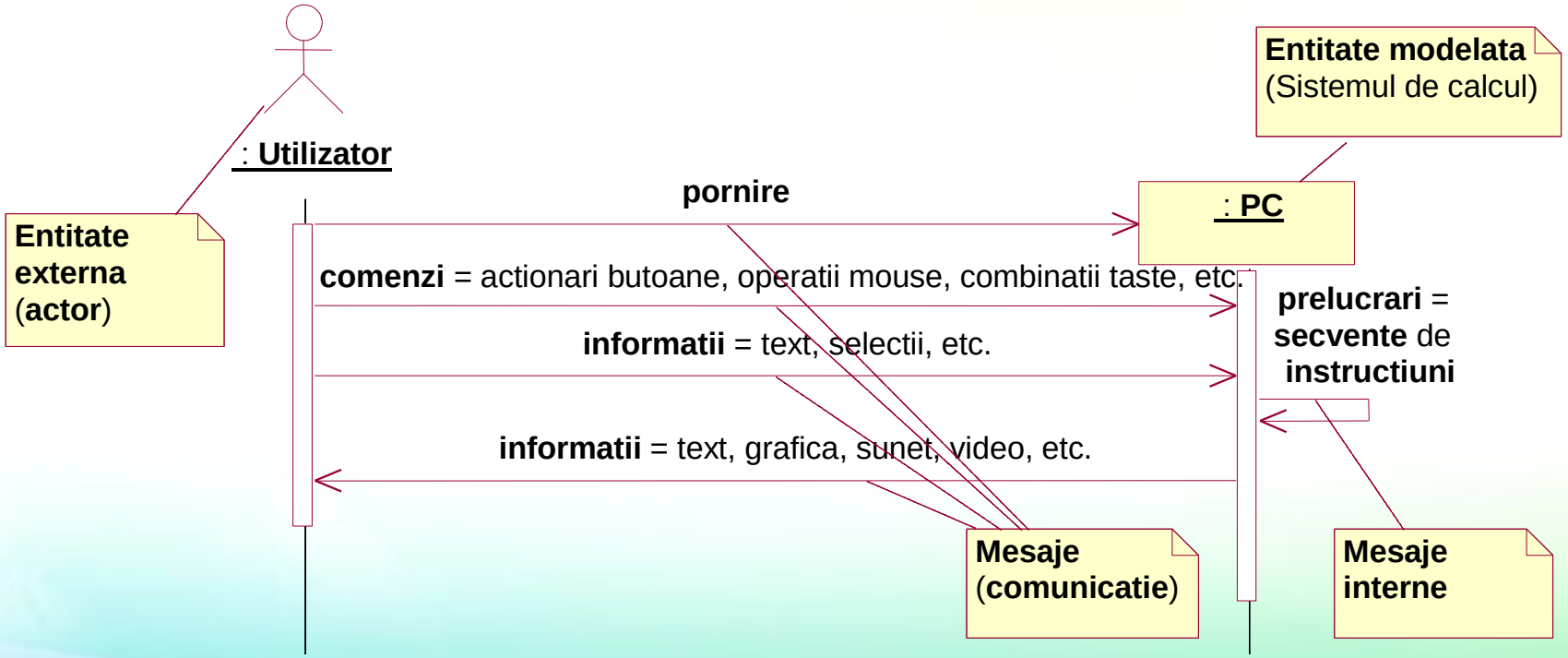
- specificatiile pot fi realizate si intr-o **forma vizuala, grafica**
- de exemplu, prin urmatoarea **diagrama MSC** (*Message Sequence Chart*)
  - care face parte si din limbajul de modelare unificat (**UML**)



Programele de calcul si masinile programabile

Pseudo-codul **echivalent** diagramei MSC

- Programul este **pornit**
- Utilizatorul **da comenzi** (actionari butoane, operatii mouse, combinatii taste, etc)
- Utilizatorul **introduce** (editeaza) **informatii** (prin consola intrare, interfata grafica, etc.)
- Programul **realizeaza** o secventa de actiuni (**prelucrari**) asupra informatiilor
- Programul **afiseaza informatii** (prin consola intrare, interfata grafica, etc.)

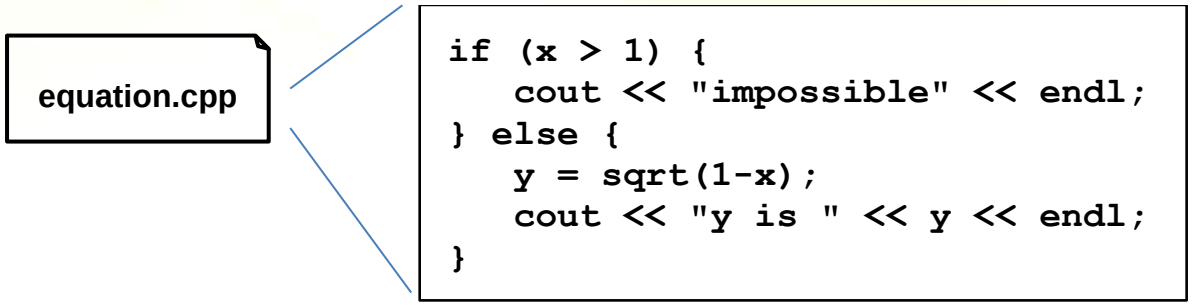


Programele de calcul si masinile programabile

Sarcinile realizate de **programele de calcul** le separa pe acestea in **categorii**:

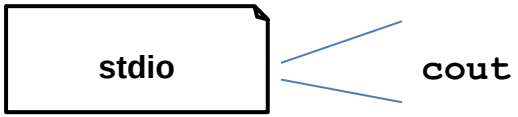
– programe **de aplicaţie** diverse

– se adreseaza direct utilizatorilor (ele sunt **tinta noastra principala**)



– programe **utilitare** (biblioteci, etc.)

– pentru programatorii care dezvolta programe de aplicatie (**pot fi tinta**)



– programe **sistem**, de **gestiune a resurselor** *hardware* si *software* (sistemele de operare/executie, in general **nu sunt tinta** noastra)

Programele de calcul si masinile programabile

Programarea (implementarea)

– este activitatea de **concepere, scriere si depanare** a programelor de calcul

– este **etapa centrala** a unui **proces mai larg**

– numit **dezvoltarea** programelor (**programarea in sens larg**)

– care cuprinde

– **analiza** (a cerintelor si a domeniului problemei)

– **proiectare** (a solutiei, arhitectural si de detaliu)

– **implementare** (codare si depanare a unitatilor de cod ale **solutiei**)

– **integrare** (a subsistemelor, unitatilor de cod, etc., in sistem)

– **testare** (a unitatilor de cod – unitara, de integrare, de ansamblu)

Cum se numeau dezvoltatorii de programe in anii ‘80-’90?

Programarea

- Programarea ca **rezolvare de probleme**
  - curs **Inginerie *Software* (?)**

– Ce **perspective** pot exista in privinta programelor?

– Paralela cu “***problem solving***” (din matematica, strategie,etc.)

– Drumul de la **cerinte** ale unei **probleme** la **solutia concreta**

– Rolurile **dezvoltatorilor** si **utilizatorilor**

– Conceptul de **cutie neagra**

– Conceptul de cutie transparenta

Programele de calcul si masinile programabile

Programele de calcul

- sunt **secvente** de **instrucţiuni** (pasi) care **conduc acţiunile** unei **maşini** programabile

secventa de pasi (instructiuni)

```
if (x > 1) {
   cout << "impossible" << endl;
} else {
   y = sqrt(1-x);
   cout << "y is " << y << endl;
}
```

- pot fi definite diferit din diferite **perspective**
  - a **programatorului** – o **solutie** la o **problema** pe care o are de rezolvat
  - a **utilizatorului** – un **serviciu** care ii este **oferit** de catre calculator

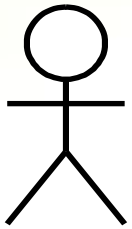
    (serviciu **creat** de catre **programator**)

Programarea ca rezolvare de probleme

Problemele de rezolvat prin programare

– sunt **enuntate** (specificate informal) de catre viitorii **utilizatori** ai programelor

– sub forma **cerintelor** privind viitoarele **servicii** oferite de programe



Utilizator (client)

Cerinte ↓

Problema reala

Enunt

daca patratul unui numar este egal cu diferenta dintre 1 si un al doilea numar, sa se afle cum depinde primul numar de al doilea

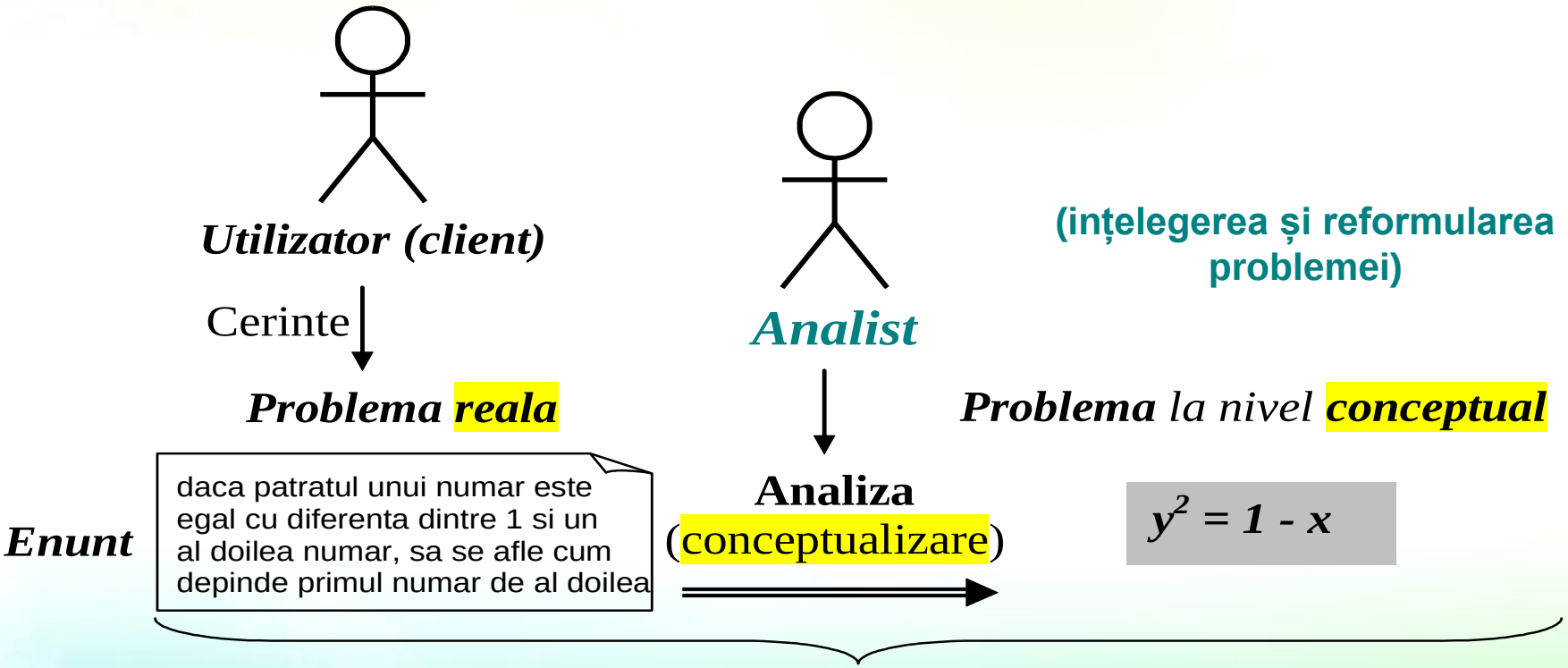
Domeniul *(contextul): matematica*

Programarea ca rezolvare de probleme

Problemele de rezolvat prin programare

– trebuie mai intai cat mai bine **analizate** si intelese in **context** (domeniu)

– fiind **transformate** din **lumea reala** in **concepte** ale domeniului



Programarea ca rezolvare de probleme

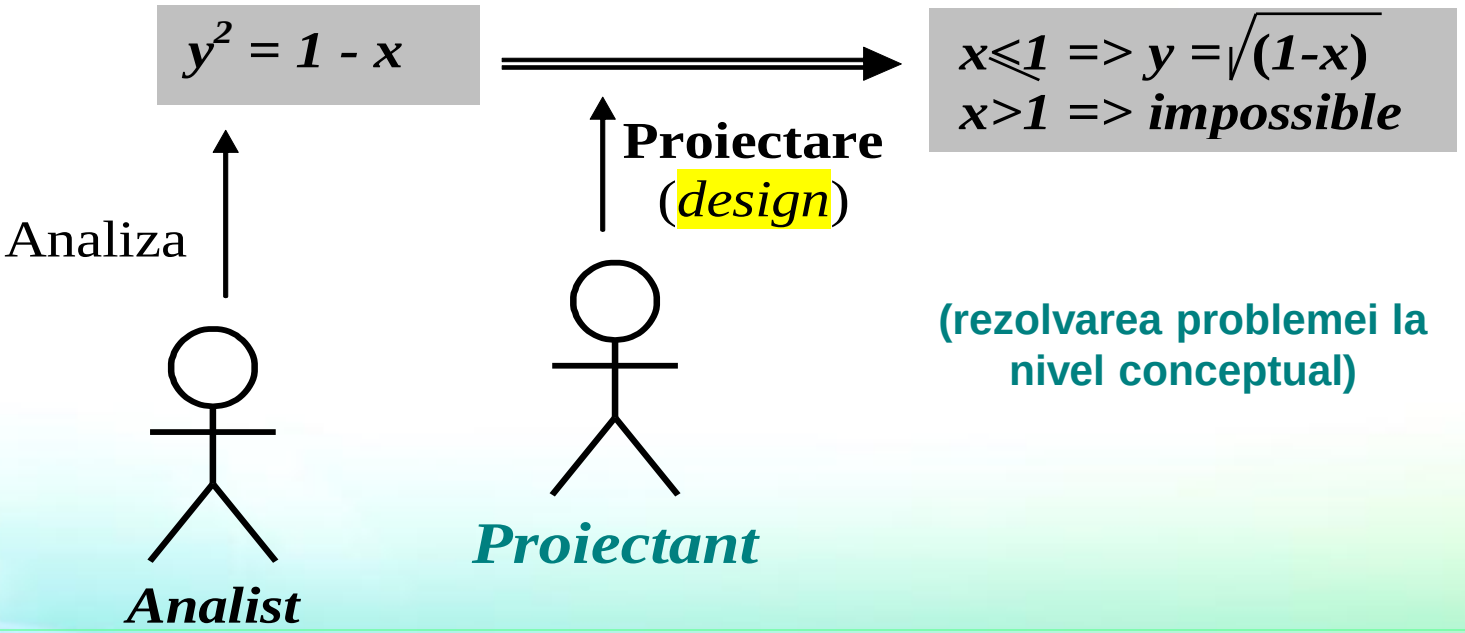
Problemele de rezolvat prin programare

– sunt apoi **rezolvate** la nivel **conceptual**

– prin **proiectarea** unei **solutii**

abstractii

Problema *la nivel* ***conceptual*** — ***Solutia*** *la nivel* ***conceptual***



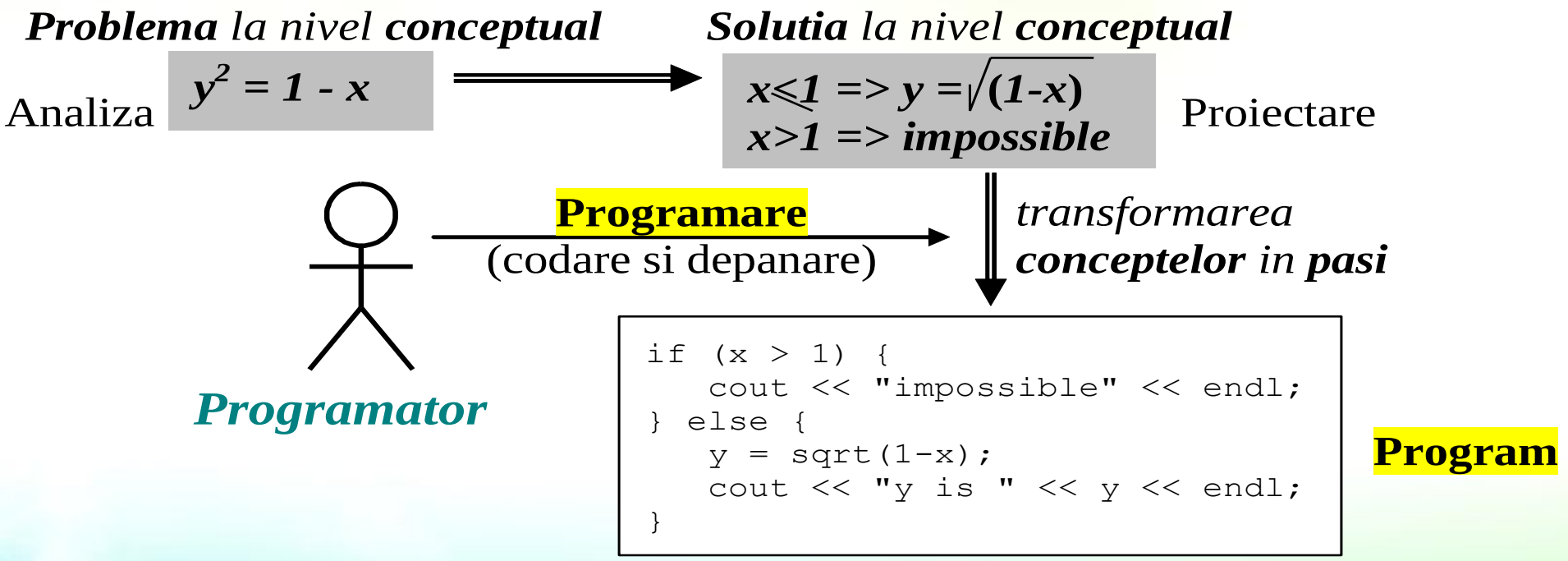
(rezolvarea problemei la nivel conceptual)

Programarea ca rezolvare de probleme

Solutia conceptuala a unei **probleme** de rezolvat prin programare

– este apoi **transformata** in pasi (instructiuni)

– care formeaza un **program** de calcul



Programarea ca rezolvare de probleme

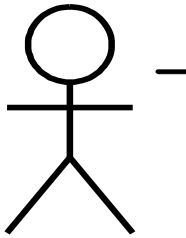
Aceasta este perspectiva programatorului

– cu diferitele sale **roluri**: analist, proiectant, programator propriu-zis

– cea **din interiorul sistemului *software*** (***inside the box***)

Problema** la nivel **conceptual — ***Solutia** la nivel **conceptual***

Analiza: $y^2 = 1 - x$ ⟹ $x \leqslant 1 \Rightarrow y = \sqrt{(1-x)}$; $x > 1 \Rightarrow$ ***impossible*** — Proiectare



Programator

Programare (codare si depanare)

transformarea ***conceptelor** in **pasi***

```
if (x > 1) {
   cout << "impossible" << endl;
} else {
   y = sqrt(1-x);
   cout << "y is " << y << endl;
}
```

Program

Inside the box **(programul)**

***Solutia** translatata **in limbaj** de programare*

Programarea ca rezolvare de probleme

Perspectiva utilizatorului

– cea **din exteriorul sistemului *software*** (***outside the box***)

Program *in limbaj de programare*

```
if (x > 1) {
   cout << "impossible" << endl;
} else {
   y = sqrt(1-x);
   cout << "y is " << y << endl;
}
```

Black Box
(abstractizari)
"cutie" care are "vizibile" doar intrarile (linii comanda) **si iesirile** (rezultatele afisarii)

Outside the box
(lumea reala)

compilare ⇓ *(translatie)*

```
> equation.exe 0
 y is 1
> equation.exe 2
 impossible
>
```

proces *(program in executie)*

```
2B 34 6F FE
24 4C 66 8A
B3 E2 78 9A
CD B1 85 32
99 04 71 00
00 00 00 00
```

Utilizator

Serviciu *oferit prin* ***interfata*** *ca urmare a* ***executiei***

cod executabil

Programarea ca rezolvare de probleme

Perspectiva utilizatorului

– **sistemul *software*** este vazut ca un **serviciu** oferit de catre o "**cutie neagra**" (***black box***)

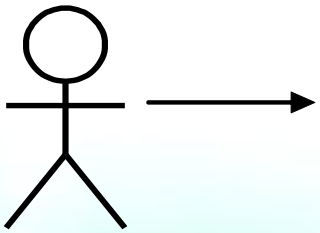
Program *in limbaj de programare*

```
if (x > 1) {
   cout << "impossible" << endl;
} else {
   y = sqrt(1-x);
   cout << "y is " << y << endl;
}
```

Black Box
(abstractizari)
"**cutie" care are "vizibile" doar intrarile** (linii comanda) **si iesirile** (rezultatele afisarii)

Outside the box
(lumea reala)

compilare ⇓ *(translatie)*



Utilizator

```
> equation.exe 0
 y is 1
> equation.exe 2
 impossible
>
```

Serviciu *oferit prin* ***interfata***
ca urmare a ***executiei***

proces *(program in executie)*

```
2B 34 6F FE
24 4C 66 8A
B3 E2 78 9A
CD B1 85 32
99 04 71 00
00 00 00 00
```

cod executabil

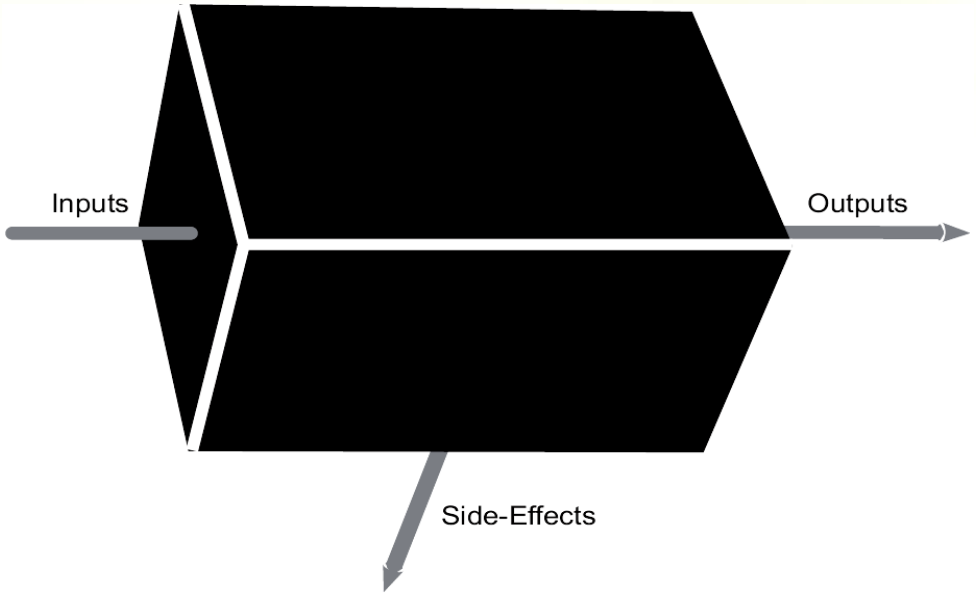
Programarea ca rezolvare de probleme

Perspectiva utilizatorului – **programul** vazut ca **serviciu** oferit de un ***black box***

Black Box

(***incapsulare, abstractizare, ascunderea detaliilor***)

Utilizator



“se vad” doar intrarile, iesirile, si efectele colaterale (*serviciile furnizate, interfetele oferite si necesare*)

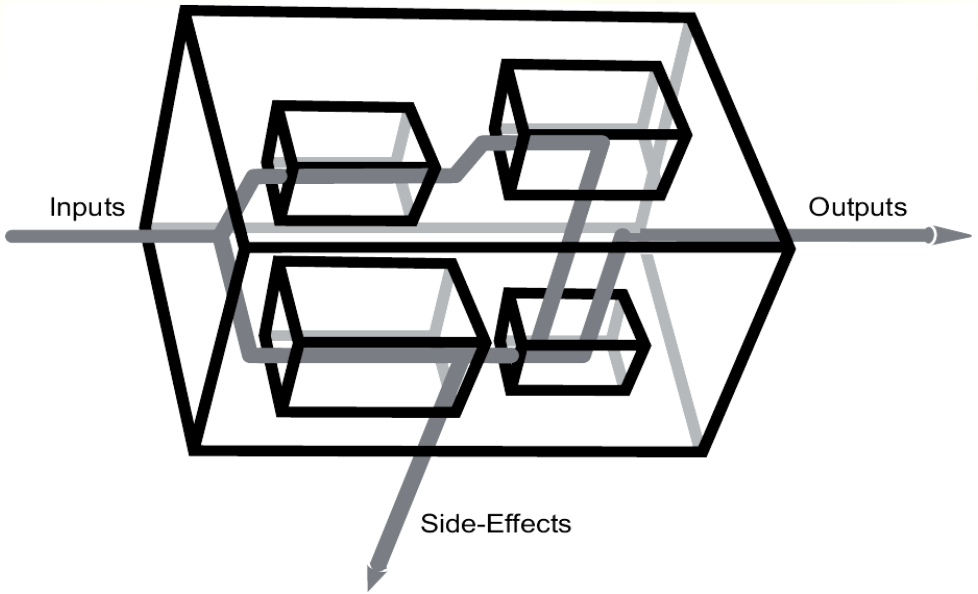
Programarea ca rezolvare de probleme

Perspectiva programatorului – **programul** vazut ca **solutie** ***white / glass box***

White / Glass Box

(***incapsulare, abstractizare***)

Programator



“se vad” si componentele,
si interactiunile dintre ele
(*detaliile de implementare*)

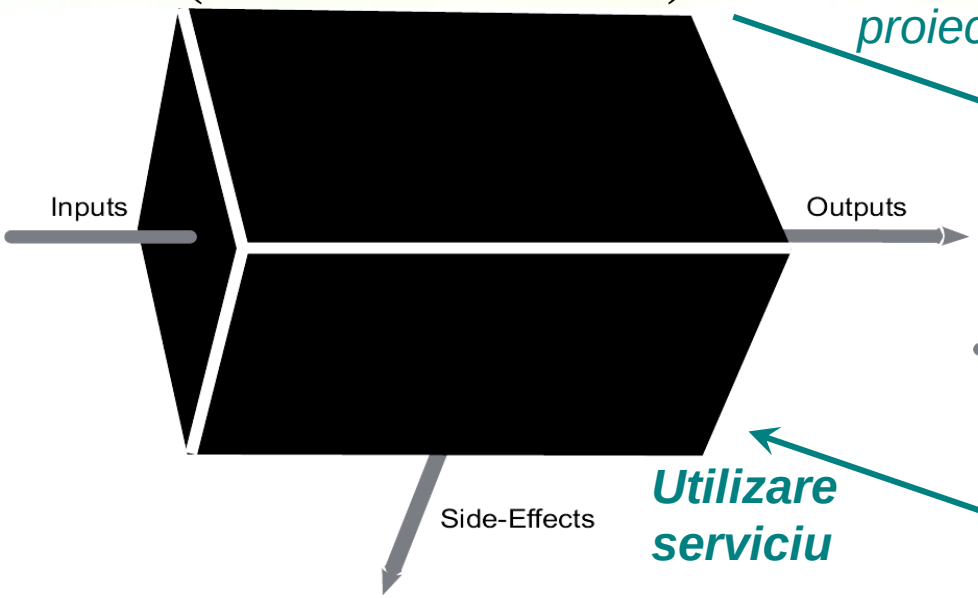
Programarea ca rezolvare de probleme

Perspective asupra programului

Utilizator

Black Box

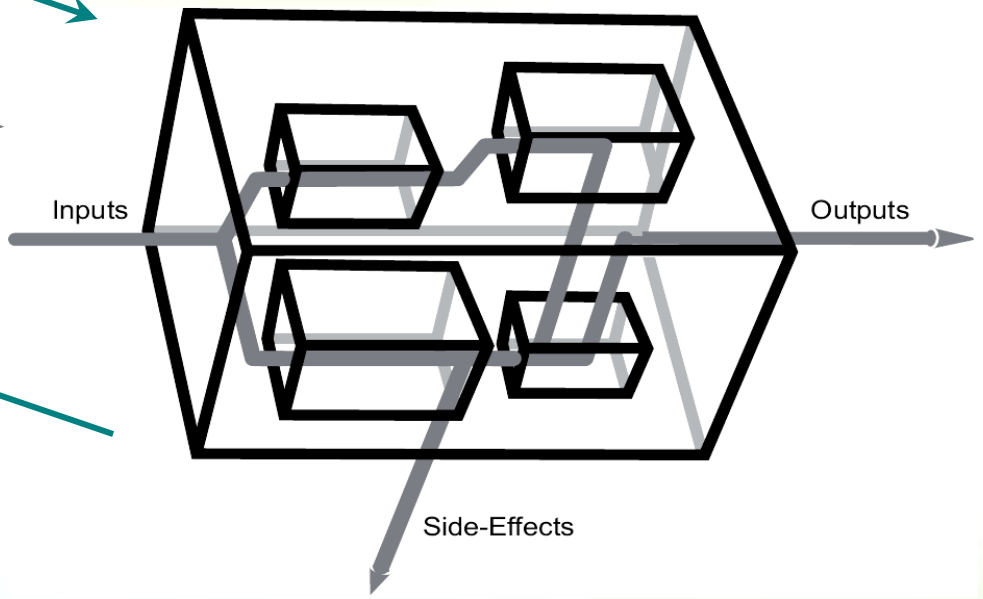
(*ascunderea detaliilor*)



***Dezvoltare (programare in sens larg:** analiza, proiectare, implementare)*

White / Glass Box

Programator



Utilizare serviciu

"se vad" doar intrarile, iesirile, si efectele colaterale (*serviciile furnizate, interfetele oferite si necesare*)

"se vad" si componentele, si interactiunile dintre ele (*detaliile de implementare*)

Programarea ca rezolvare de probleme

Programele de calcul (sistemele *software*)

– «*sunt **solutii** ale unor **probleme** **de rezolvat** intr-un **context** **dat***»

- **contextul** = **domeniul** problemei si **constrangerile**
- **problema** = **cerintele** utilizatorului si ale aplicatiei
- **rezolvarea** = **procesul de dezvoltare** (analiza, proiectare, codare)
- **subsolutii** = diagrame, **modele**, prototipuri, **subsisteme**, unitati de cod
- **solutia** = **produsul final** (programul)

Matematica

daca patratul unui numar este
egal cu diferenta dintre 1 si un
al doilea numar, sa se afle cum
depinde primul numar de al doilea

$y^2 = 1 - x$ ⇒ $x \leq 1 \Rightarrow y = \sqrt{(1-x)}$; $x > 1 \Rightarrow$ *impossible* ⇒

```
if (x > 1) {
   cout << "impossible" << endl; } else
{
```

```
> equation.exe 0
 y is 1
```