PROGRAMARE ORIENTATĂ PE OBIECTE

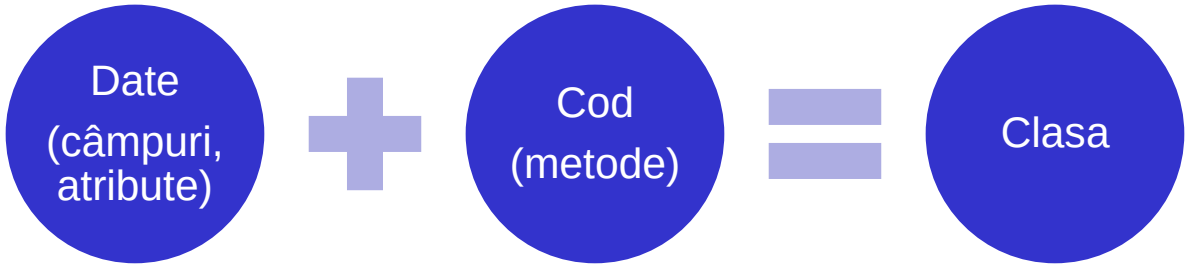
GENGE BÉLA

Capitolul 2
Clase și Obiecte

Clase

Clasa:

- Asigură încapsularea unei entități.
- Reprezintă un tip de dată definit de utilizator, asigură o descriere a unui concept.



Exemplu – programare procedurală

Încapsularea conceptului “Termostat” în C:

```
1  struct Termostat {
2      double tcam;
3      double tdorit;
4  };
5
6  typedef struct Termostat* TPTR;
7
8  TPTR create_termostat(double tcam, double tdorit);
9  void increase_targett(TPTR t);
10 void decrease_targett(TPTR t);
```

Exemplu – programare OO

Încapsularea conceptului “Termostat” în Java:

```
1  public class Termostat {
2      private double _tcam;
3      private double _tdorit;
4
5      public Termostat(double tcam, double tdorit) {
6          ...
7      }
8
9      public void increase_targett() {
10         ...
11     }
12
13     public void decrease_targett() {
14         ...
15     }
16 }
```

Câmpuri

Metode

Definiții de bază

Modificatori de acces:

- Definesc vizibilitatea claselor, câmpurilor și metodelor.
- public, private, protected, implicit.

	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

Definiții de bază

Obiectul:

- Reprezintă o instanță a clasei.
- Instanțierea returnează o referință.
- Instanțierea clasei termostat:

```
1  Termostat t = new Termostat(0, 25);
```

Definiții de bază

Pentru două instanțe diferite:

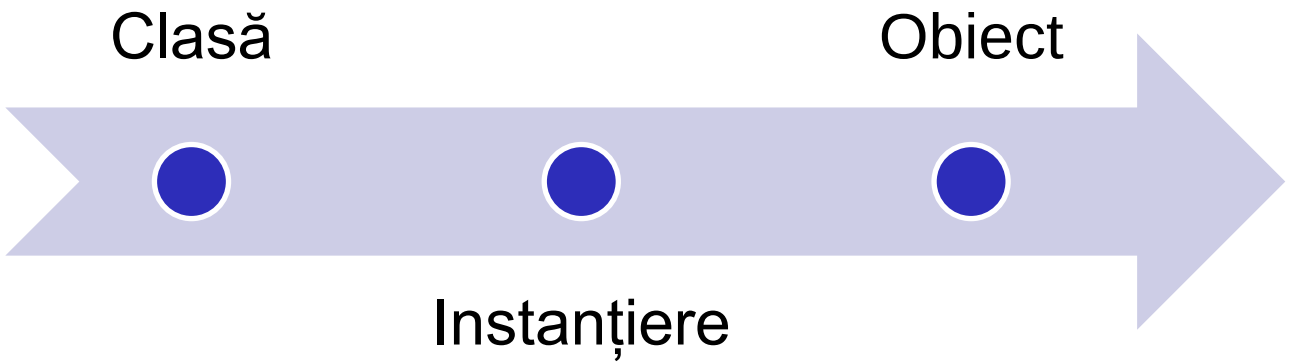
- Metodele se aplică asupra datelor din instanța identificată prin referință.

```
1  Termostat t1 = new Termostat(5, 25);
2  Termostat t2 = new Termostat(5, 30);
3
4  t1.increase_targett();
5  t2.decrease_targett();
```

Definiții de bază

Instanțierea:

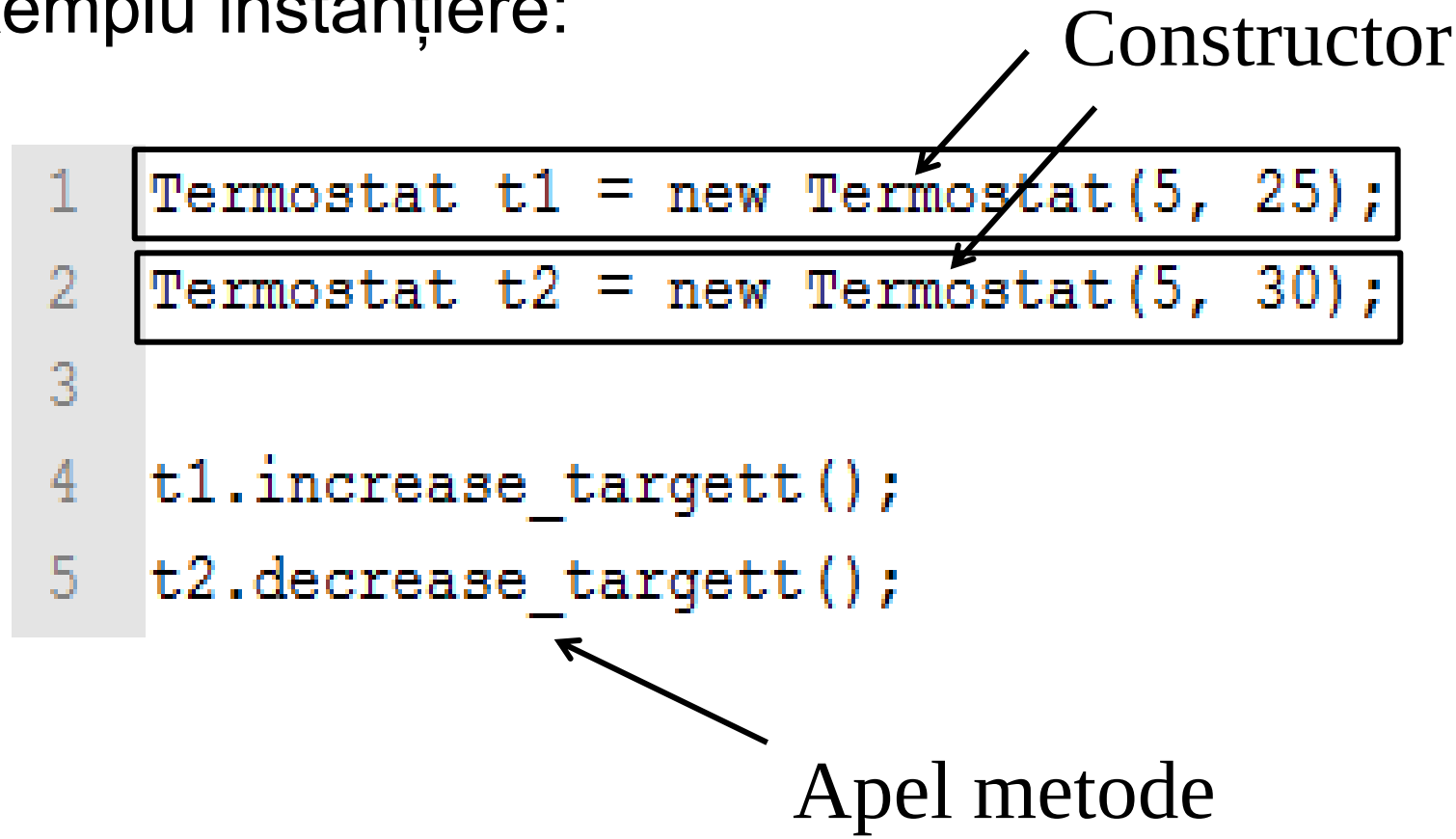
- Se aplică asupra unei clase cu scopul creării unui obiect.



- Pașii instanțierii:
  - Alocare memorie (heap): operatorul **new**.
  - Apel metodă specială de inițializare: **CONSTRUCTORUL**.
  - Returnarea **referinței** către obiectul nou creat.

Definiții de bază

Exemplu instanțiere:



Definiții de bază

Constructorul:

- Metodă specială cu rolul de inițializare.

Proprietățile constructorului:

- Numele identic cu numele clasei.
- Nu are tip returnat.
- Poate avea asociat modificatori de acces.
- O clasă poate avea mai mulți constructori diferențiați prin tipul și numărul de parametrii.
- Dacă nu se definește un constructor se generează unul automat – **constructor implicit**

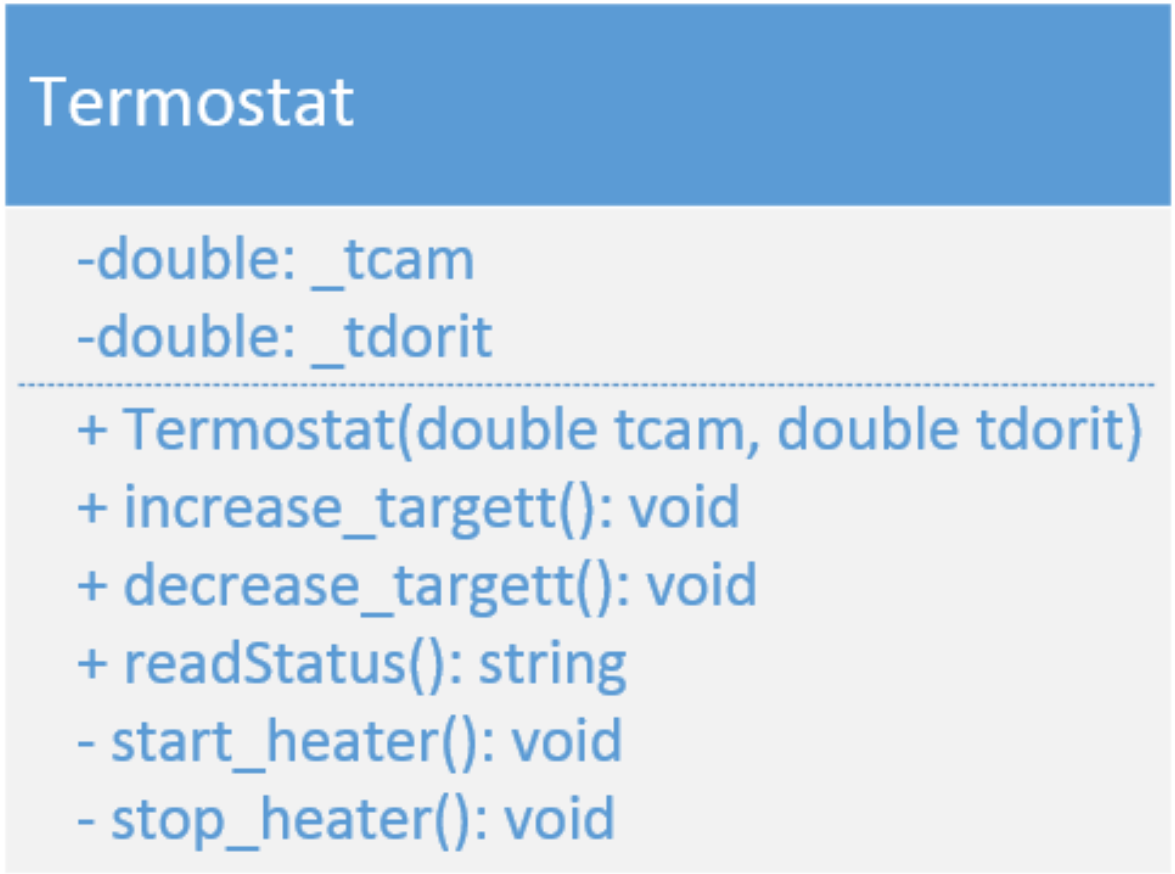
Definiții de bază

Proprietățile constructorului implicit:

- Nu are parametrii.
- Câmpurile se vor inițializa cu valoarea implicită.
- Doar acel constructor este numit **implicit** care se generează de compilator -> constructorul definit de utilizator cu același format ca și constructorul implicit NU este constructor implicit.

Definiții de bază

Exemplu diagrama de clasă:

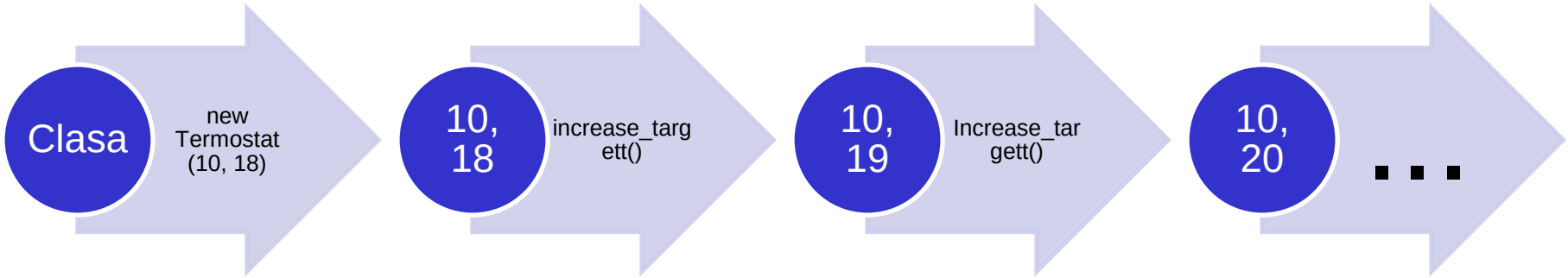


Starea obiectelor

Starea:

- Este dată de setul valorilor câmpurilor.
- Se modifică prin apelul metodelor.

Exemplu de modificare stare:



Referința “this”

Reprezintă:

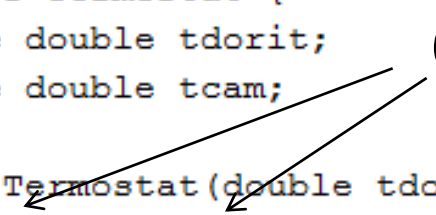
- “Adresa” obiectului curent.

- Permite referențierea metodelor și câmpurilor instanței curente.

Referința “this”

Exemplu de problemă distincție câmpuri și parametrii:

```
1  public class Termostat {
2      private double tdorit;
3      private double tcam;
4
5      public Termostat(double tdorit, double tcam) {
6          tdorit = tdorit;
7          tcam = tcam;
8      }
```



Posibilă soluție:

```
1  public class Termostat {
2      private double tdorit;
3      private double tcam;
4
5      public Termostat(double tdorit, double tcam) {
6          this.tdorit = tdorit;
7          this.tcam = tcam;
8      }
```

Distrugerea obiectelor

Aspecte specifice Java:

- Se realizează automat de **Garbage Collector (GC)**.
- Pentru fiecare obiect se menține o tabelă cu numărul de referințe.
- Când numărul de referințe devine 0 obiectul este planificat pentru distrugere.
- GC rulează pe un fir de execuție separat.

Aspecte specifice altor limbaje (C++):

- **delete** și **destructor**

Exercițiu: pseudo-cod pentru GC.

Distrugerea obiectelor

Metoda finalize() din Java:

- Implementarea nu este obligatorie.
- Momentul exact al apelului nu este cunoscut.
- Poate fi folosită pentru eliberarea resurselor – cu toate că este mai indicată definirea unei metode dedicate, apelată deterministic de utilizator.
- Descrierea metodei:

```
1  protected void finalize() throws Throwable;
```

Metode get/set

- Proprietatea de încapsulare ascunde implementarea:
  - Accesul la câmpuri este restricționat.
- Soluția:
  - Metode de tip getter/setter (get/set).
- Exemplu:
  - public String getName();
  - public void setName(String name);

Exercițiu

- Să se încapsuleze entitatea Persoana, cu următoarele detalii:
  - Nume, Prenume, CNP, anul nașterii, luna nașterii, ziua nașterii.
  - Adresa.
- Să se definească:
  - mai mulți constructori.
  - Metode get/set.
  - Metoda toString().

Metode și câmpuri de clasă/instanță

Două tipuri principale:

- Metode și câmpuri de instanță.
- Metode și câmpuri de clasă.

Metode/câmpuri de instanță:

- Sunt specifice instanțelor claselor.
- Pot fi utilizate (accesate, apelate) doar cu condiția existenței unei instanțe.
- În exemplul Termostat toate câmpurile și metodele sunt de instanță.

Metode și câmpuri de clasă/instanță

Metode/câmpuri de clasă:

- Sunt specifice claselor, pot fi utilizate în afara instanțelor.
- Au fost introduse deoarece Java nu include funcții în afara claselor.
- Există situații când e necesar apelul metodelor fără instanțierea clasei (vezi exemplul cu main()).
- Se declară prin cuvântul cheie **static**.
- Se apelează prin numele clasei.

Metode și câmpuri de clasă/instanță

Exemplu definire metode de clasă:

```
1  public class OpMatematice {
2      public static double diff (double a, double b) {
3          return a-b;
4      }
5      public static boolean estePrim(int a) {
6          boolean result = false;
7
8          ...
9          return result;
10     }
11 }
```

Apel:

```
1  OpMatematice.diff(10, 12);
2  OpMatematice.estePrim(11);
```

Metode și câmpuri de clasă/instanță

Exemplu apel println():

```
4   System.out.println("Rezultat: " + OpMatematice.diff(10, 12));
```

Exemplu numărarea instanțelor:

```
1   public class NumInst {
2
3       private static int _num = 0;
4
5       public NumInst() {
6           _num++;
7       }
8
9       public static int getNumInst() {
10          return _num;
11      }
12  }
```

```
14  NumInst i1 = new NumInst();
15  NumInst i2 = new NumInst();
16  NumInst i3 = new NumInst();
17
18  System.out.println(NumInst.getNumInst());
19  System.out.println(i2.getNumInst());
20
```

Metode și câmpuri de clasă/instanță

Design pattern: Clase Singleton.

Exemplu de proiectare a unei clase de tipul singleton pentru scrierea string-urilor intr-un fisier.

Scrierea în fișiere:

```
import java.io.*;
File f = new File(“./logf.txt");
if (!f.exists()) { file.createNewFile(); }
FileWriter fw = new FileWriter(f.getAbsoluteFile());
BufferedWriter bw = new BufferedWriter(fw);
bw.write(mesaj);
bw.close();
```

Metode și câmpuri de clasă/instanță

Design pattern: Clase Singleton.

Exemplu de proiectare a unei clase de tipul singleton pentru scrierea string-urilor intr-un fisier.

Crearea unei mărci de timp:

```
import java.util.Date;

String timeStamp = new SimpleDateFormat("yyyy.MM.dd
HH.mm.ss").format(new Date());


StringBuffer out = timeStamp;
out.append(“ ”);
out.append(mesaj);
```

Puncte de intrare și ieșire

Punctul de intrare:

- Metoda main().
- De regulă clasa în care e definit main() nu se instanțiază.

Terminarea execuției:

- Revenirea din metoda main().
- ... sau la apelul funcției System.exit()

Transferul parametrilor

Parametrii sunt transferați prin valoare:

- Nu se pot aduce modificări persistenete asupra variabilelor scalare.
- Modificările sunt însă persistente la obiecte prin transferul referințelor.
- Dacă dorim modificarea unei valori scalare:
  - Returnarea valorii.
  - Crearea unui obiect vector cu un singur element și transmiterea referinței ca parametru.

Transferul parametrilor

Exemplu:

- Modificarea valorii unui întreg.
- Modificarea valorii unui String transmis ca parametru (noua valoare = “”).
- Atenție! A se folosi StringBuffer în locul String dacă se doresc modificări.
  - StringBuffer: metoda delete(int start, int end).

Clasa Scanner

- Clasa ce permite procesarea tipurilor de date primitive introduse de la tastatură.
- Definit în **java.util**;
- Metode principale:
  - nextInt(): citește următorul întreg.
  - next(): citește următorul string.
  - nextDouble(): citește următorul double.
  - nextFloat(): citește următorul float.
- Exemplu:
  - import java.util.* sau import java.util.Scanner;
  - Scanner s = new Scanner(System.in);
  - int a = s.nextInt();
  - String str = s.next();

Exerciții de încapsulare

- Obiectul Stivă.
- Obiectul Mesaj într-o comunicare multi-client cu un server (Yahoo, Skype).