

Programare Orientată pe Obiecte

Clase & Obiecte

Dr. Petru Florin Mihancea

V2018092

Definiții

O *clasă* definește un set de obiecte care au aceeași structură și comportament

Un obiect este o instantă a unei clase

Booch - OO Analysis and Design

Dr Petru Florin Mihăescu

1

Diferență între clasă și obiect ?

Dr. Botan: Florin Mihăescu

Clasa vs. Obiect



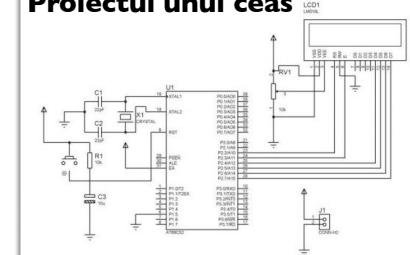
<http://www.timetools.co.uk>



Objekte



www.associazionedigitale.it



http://hubpages.com/hub/digital_clock_using_microcontroller_89C5288S

Dr Petru Florin Mihaiță

Clasa

**Implementarea unui obiect este
dată/definită de clasa acelui
obiect. Clasa specifică datele
interne ale unui obiect și
reprezentarea lor și definește
operatiile obiectului**

GOE 1995

Definiție (parțială)

Programarea orientată pe obiecte
este o metodă de **implementare a programelor**
în care acestea sunt organizate ca și
colecții de obiecte ce cooperează între ele [...],
fiecare obiect fiind o instanță a unei clase

Booch - OO Analysis and Design

Dr. Petru Florin Mihancea

2

Simulare în C

Cum ar fi în limbajul C (I) ?

C nu este object-oriented
dar putem "simula"

```
typedef struct _Clock {  
    int hour, minute, seconds;  
} Clock;  
  
void setTime(Clock *this, int hour, int minute, int second) {  
    this->hour = ((hour >= 0 && hour < 24) ? hour : 0);  
    this->minute = ((minute >= 0 && minute < 60) ? minute : 0);  
    this->seconds = ((second >= 0 && second < 60) ? second : 0);  
}  
  
void print(Clock *this) {  
    printf("Current time %d:%d:%d\n", this->hour, this->minute,  
          this->seconds);  
}
```

~ un fel de clasă

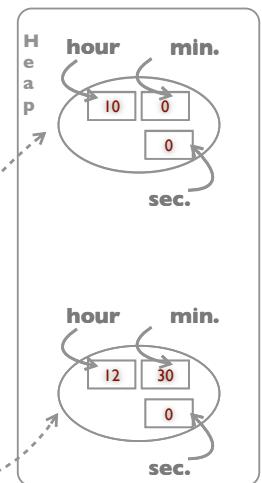
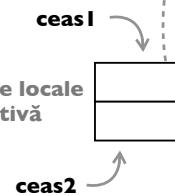
Dr. Petru Florin Mihancea

```
int main(int argc, char** argv) {  
    Clock *ceas1 = malloc(sizeof(Clock));  
    Clock *ceas2 = malloc(sizeof(Clock));  
    setTime(ceas1, 10, 0, 0);  
    print(ceas1);  
    setTime(ceas2, 12, 30, 0);  
    print(ceas2);  
    print(ceas1);  
    ...  
    //Eliberare memorie  
    ...  
}
```

Console
Current time 10:0:0
Current time 12:30:0
Current time 10:0:0

Cum ar fi în limbajul C (II) ?

Variabile locale
pe stivă



Dr. Petru Florin Mihancea

3

Clase și Obiecte în Java

Dr. Petru Florin Mihancea

Variabile/câmpuri instanță

```
class Clock {  
    //Modificatori tip nume1, nume2 , ... , numeN = initVal;  
    private int hour, minute, seconds;  
    ...  
}
```

definesc reprezentarea datelor unui obiect

fiecare obiect are propriul exemplar (locație de memorie) pt. fiecare din variabilele sale instanță

Dr. Petru Florin Mihancea

Ceasuri în Java

```
class Clock {  
  
    private int hour, minute, seconds;  
  
    public void setTime(int h, int m, int s) {  
        hour = (h >= 0) && (h < 24) ? h : 0;  
        minute = (m >= 0) && ( m < 60) ? m : 0;  
        seconds = (s >= 0) && ( s < 60) ? s : 0;  
    }  
  
    public void print() {  
        System.out.println("Current time " + hour + ":" + minute + ":" + seconds);  
    }  
}
```

Dr. Petru Florin Mihancea

Metode instanță

```
class Clock {  
    ...  
    //Modificatori tip_returnat nume(tip param1, ... , tip paramN) ...  
    public void setTime(int h, int m, int s) {  
        hour = (h >= 0) && (h < 24) ? h : 0;  
        minute = (m >= 0) && ( m < 60) ? m : 0;  
        seconds = (s >= 0) && ( s < 60) ? s : 0;  
    }  
    ...  
}
```

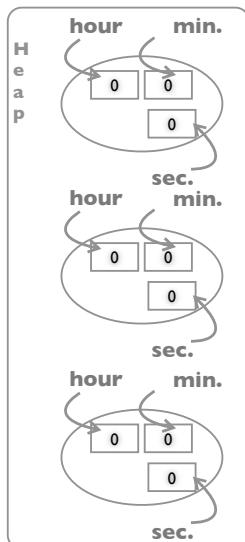
definesc comportamentul / operațiile ce se pot executa pe acel obiect

Dr. Petru Florin Mihancea

Crearea obiectelor

la rularea programului prin operatorul **new** și sunt alocate în **heap**

```
class Main {  
    public static void main(String argv[]) {  
        ...  
        new Clock();  
        new Clock();  
        new Clock();  
        ...  
    }  
}
```



Dr. Petru Florin Mihancea

Ștergerea obiectelor

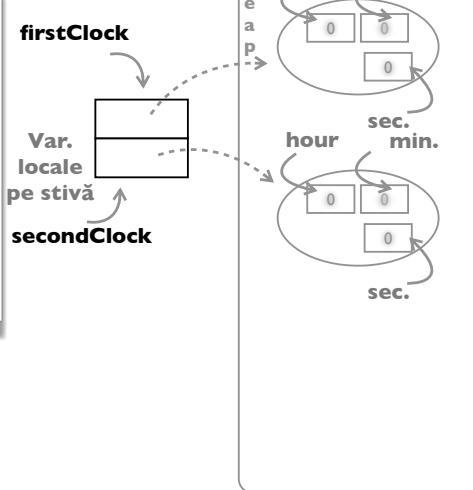
De acest lucru se ocupă colectorul de deșeuri

În esență, acesta șterge singur din timp în timp obiectele care nu mai pot fi accesate din program

Dr. Petru Florin Mihancea

Referirea obiectelor

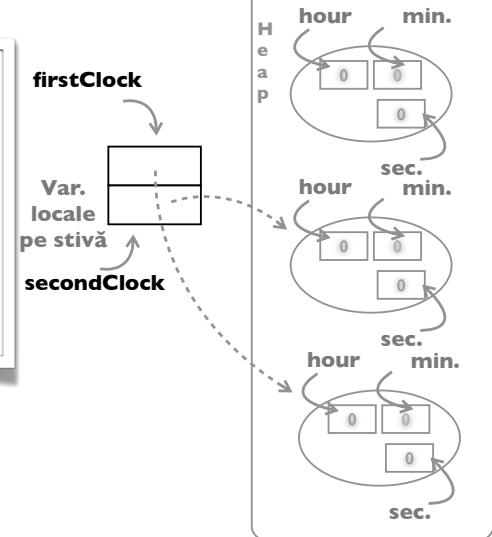
```
class Main {  
    public static void main(String argv[]) {  
        ...  
        //Variabile referință  
        //Aici variabile locale dar pot fi și câmpuri  
        //NumeClasa numeVariabila;  
        Clock firstClock, secondClock;  
        firstClock = new Clock();  
        secondClock = new Clock();  
    }  
}
```



Dr. Petru Florin Mihancea

```
class Main {  
    public static void main(String argv[]) {  
        ...  
        //Variabile referință  
        //Aici variabile locale dar pot fi și câmpuri  
        //NumeClasa numeVariabila;  
        Clock firstClock, secondClock;  
        firstClock = new Clock();  
        secondClock = new Clock();  
        firstClock = new Clock();  
    }  
}
```

Referirea obiectelor



Dr. Petru Florin Mihancea

Referirea obiectelor

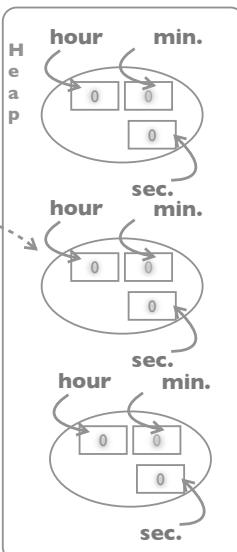
```
class Main {
    public static void main(String argv[]) {
        ...
        //Variabile referință
        //Aici variabile locale dar pot fi și câmpuri
        //NumeClasa numeVariabila;
        Clock firstClock, secondClock;
        firstClock = new Clock();
        secondClock = new Clock();
        firstClock = new Clock();
        firstClock = null;
        ...
    }
}
```

- **null** - valoare ce arată că o referință nu referă niciun obiect

- nu e echivalent cu 0 !

Dr. Petru Florin Mihăescu

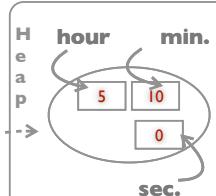
firstClock
Var. locale pe stivă secondClock



```
class Main {
    public static void main(String argv[]) {
        //operatorul . (punct)
        Clock firstClock, secondClock;
        firstClock = new Clock();
        firstClock . setTime(5, 10, 0);
    }
}
```

Invocarea operațiilor

firstClock
Var. locale pe stivă secondClock



```
class Clock {
    ...
    public void setTime(int h, int m, int s) {
        hour = (h >= 0) && (h < 24) ? h : 0;
        minute = (m >= 0) && (m < 60) ? m : 0;
        seconds = (s >= 0) && (s < 60) ? s : 0;
    }
    ...
}
```

Dr. Petru Florin Mihăescu

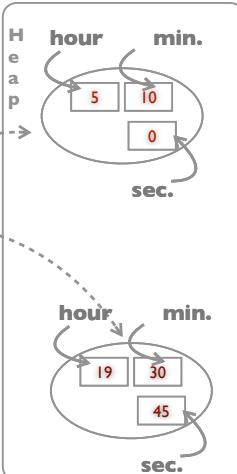
Invocarea operațiilor

```
class Main {
    public static void main(String argv[]) {
        //operatorul . (punct)
        Clock firstClock, secondClock;
        firstClock = new Clock();
        firstClock . setTime(5, 10, 0);
        secondClock = new Clock();
        secondClock . setTime(19, 30, 45);
        ... // tot . (punct) și acces la câmpuri
    }
}
```

```
class Clock {
    ...
    public void setTime(int h, int m, int s) {
        hour = (h >= 0) && (h < 24) ? h : 0;
        minute = (m >= 0) && (m < 60) ? m : 0;
        seconds = (s >= 0) && (s < 60) ? s : 0;
    }
    ...
}
```

De unde știe pe câmpurile cărui obiect lucrează ?

firstClock
Var. locale pe stivă secondClock

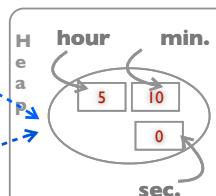


```
class Clock {
    ...
    public void setTime(int h, int m, int s) {
        this.hour = (h >= 0) && (h < 24) ? h : 0;
        this.minute = (m >= 0) && (m < 60) ? m : 0;
        this.seconds = (s >= 0) && (s < 60) ? s : 0;
    }
    ...
}
```

În timpul execuției unei metode instantă, **this** referă spre obiectul din fața apelului (obiectul receptor)

```
...
Clock firstClock, secondClock;
firstClock = new Clock();
secondClock = new Clock();
firstClock . setTime(5, 10, 0);
secondClock . setTime(19, 30, 45);
...
```

Referința this

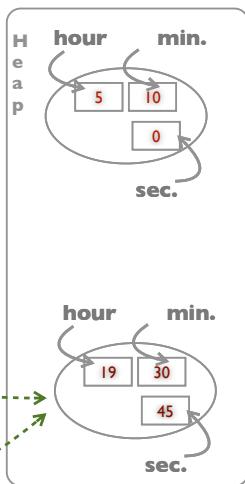


Referință this

```
class Clock {  
    ...  
    public void setTime(int h, int m, int s) {  
        this.hour = (h >= 0) && (h < 24) ? h : 0;  
        this.minute = (m >= 0) && (m < 60) ? m : 0;  
        this.seconds = (s >= 0) && (s < 60) ? s : 0;  
    }  
    ...  
}
```

în timpul execuției unei metode instantă, **this** referă spre obiectul din fața apelului (obiectul receptor)

```
...  
Clock firstClock, secondClock;  
firstClock = new Clock();  
secondClock = new Clock();  
firstClock.setTime(5, 10, 0);  
secondClock.setTime(19, 30, 45);  
...
```

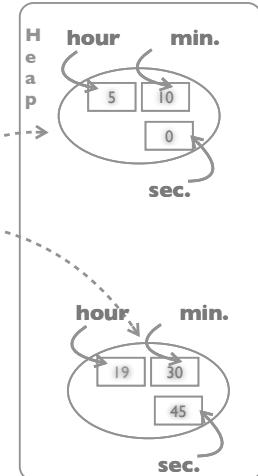


Dr. Petru Florin Mihancea

Atenție la aliasing !

```
class Main {  
    public static void main(String args[]) {  
        ...  
        Clock firstClock, secondClock;  
        firstClock = new Clock();  
        firstClock.setTime(5, 10, 0);  
        secondClock = new Clock();  
        secondClock.setTime(19, 30, 45);  
    }  
}
```

Var. locale pe stivă
firstClock

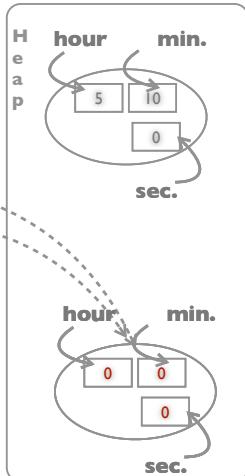


Dr. Petru Florin Mihancea

Atenție la aliasing !

```
class Main {  
    public static void main(String args[]) {  
        ...  
        Clock firstClock, secondClock;  
        firstClock = new Clock();  
        firstClock.setTime(5, 10, 0);  
        secondClock = new Clock();  
        secondClock.setTime(19, 30, 45);  
        firstClock = secondClock;  
        firstClock.setTime(0, 0, 0);  
        secondClock.print();  
    }  
}
```

firstClock
Var. locale pe stivă
secondClock



Output
Current time 0:0:0

Dr. Petru Florin Mihancea

4

Constructorii

Dr. Petru Florin Mihancea

Ce este un constructor ?

“Metode” mai speciale care

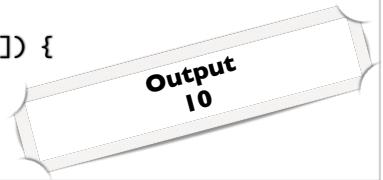
- au **exact același nume ca și clasa**
- **nu au tip returnat (nici măcar void)**
- **se poate executa o dată la crearea obiectului**

```
class Clock {  
    ...  
    public Clock(int h, int m, int s) {  
        setTime(h,m,s);  
    }  
    ...  
}
```

Dr. Petru Florin Mihancea

Ordinea inițializărilor (aprox)

```
Class Student {  
    private int numarMatricol = 0;  
    public Student(int nr) {  
        numarMatricol = nr;  
    }  
    public void print() {  
        System.out.println(numarMatricol);  
    }  
    public static void main(String argv[]) {  
        Student s1 = new Student(10);  
        s1.print();  
    }  
}
```



1. Se execută inițializatorii variabilelor instanță în ordine textuală
2. Se execută corpul constructorului

Dr. Petru Florin Mihancea

Utilizarea

```
class Exemplu {  
    public static void main(String argv[]) {  
        //...  
        Clock firstClock, secondClock, thirdClock;  
        firstClock = new Clock(10, 0, 0);  
        secondClock = new Clock(12, 30, 40);  
        thirdClock = new Clock(); //Eroare de compilare  
        //...  
    }  
}
```

Dacă nu declarăm niciun constructor în clasă se generează automat unul fără nici un argument

Un constructor fără argumente se mai numește **constructor no-arg**

Dr. Petru Florin Mihancea

5

Accesul la membrii unei clase

Modificatori de access

Dr. Petru Florin Mihancea

Modificatori/Specificatori de acces (I)

```
class Clock {  
  
    private int hour, minute, seconds;  
  
    ...  
}
```

private

respectivul membru al clasei (câmp/metodă) poate fi accesat doar în interiorul clasei

public

respectivul membru al clasei (câmp/metodă) poate fi accesat de oriunde

Atenție: dacă lipsește nu e nici public nici private (vom vedea mai încolo)

Dr. Petru Florin Mihancea

Euristică Importantă a Programării Orientate pe Obiecte

All data should be hidden (private) within its class

Riel's Heuristic 2.1

Dr. Petru Florin Mihancea

Modificatori/Specificatori de acces (II)

```
class Specifier{  
    public int publicAttribute;  
    private int privateAttribute;  
    public void publicMethod() {  
        publicAttribute = 20;  
        privateAttribute = privateMethod();  
    }  
    private int privateMethod() {  
        return 10;  
    }  
    public void otherMethod(Specifier s) {  
        publicAttribute = s.privateAttribute;  
        privateAttribute = s.privateMethod();  
        s.publicMethod();  
    }  
}  
class ClientSpecifier{  
    public static void main(String[] args) {  
        Specifier s = new Specifier();  
        s.publicMethod();  
        s.privateMethod(); //Linie cu eroare de compilare  
    }  
}
```

Dr. Petru Florin Mihancea

De ce ?

```
class A {  
    ...  
    public void method1(ComplexNumber n) {  
        ... // computation based on n.x, n.y  
    }  
    ...  
}
```

```
class D {  
    ...  
    public void method4(ComplexNumber n) {  
        ... // computation based on n.x, n.y  
    }  
    ...  
}
```

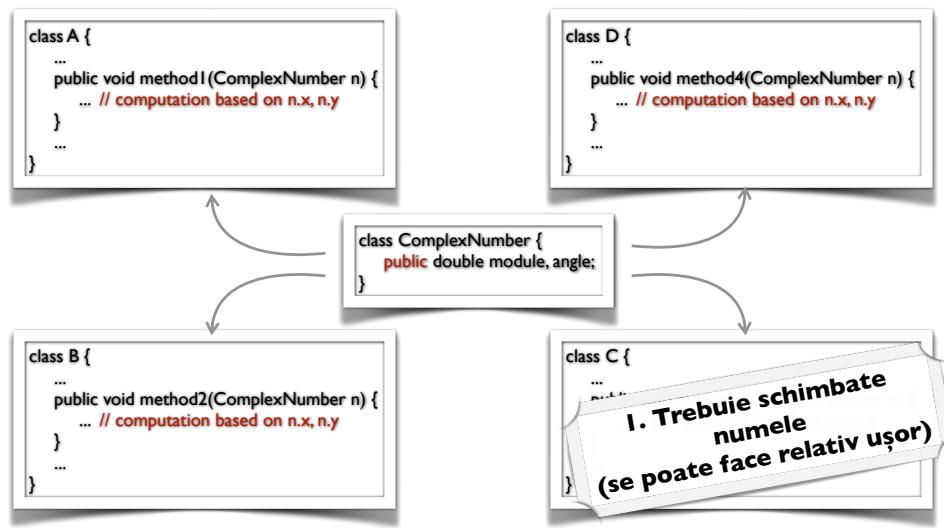
```
class ComplexNumber {  
    public double x,y;  
}
```

```
class B {  
    ...  
    public void method2(ComplexNumber n) {  
        ... // computation based on n.x, n.y  
    }  
    ...  
}
```

Constatăm la un moment dat ca ar fi mai bine să reprezentăm prin modul și unghi un număr complex

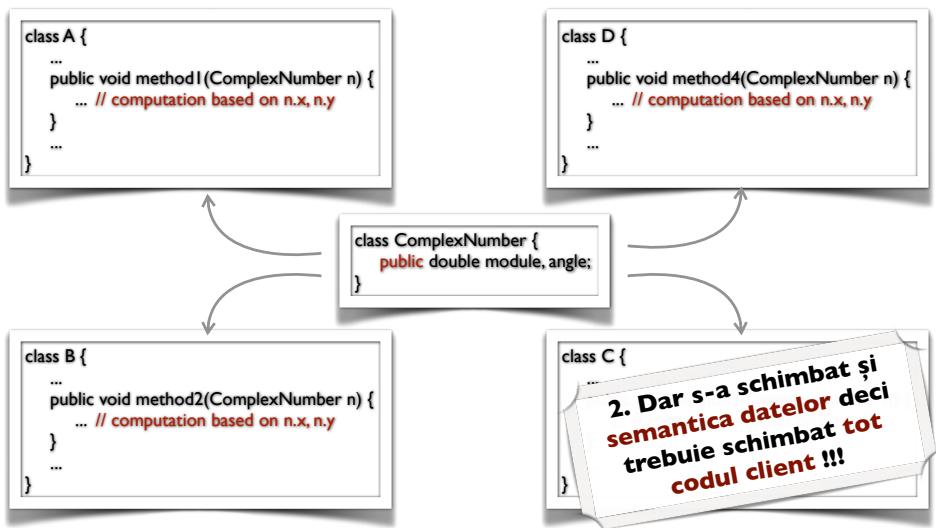
Dr. Petru Florin Mihancea

De ce ?



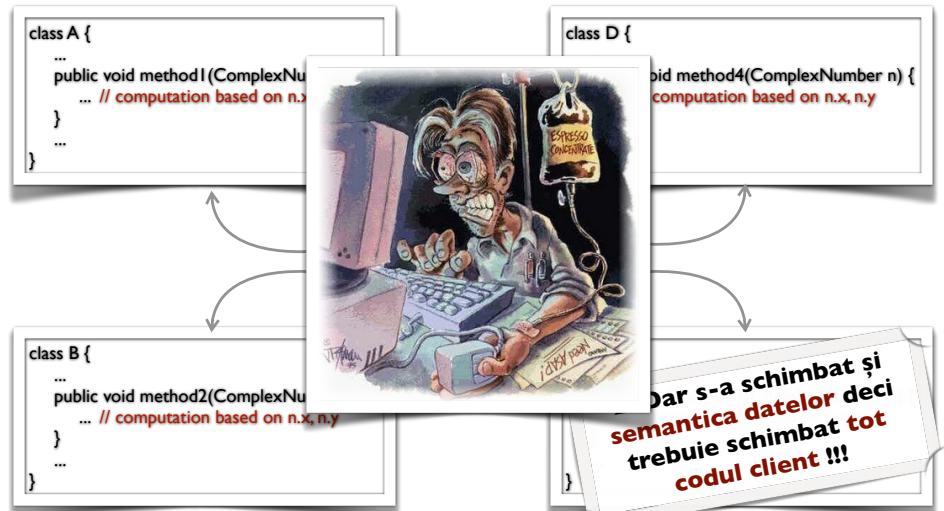
Dr. Petru Florin Mihancea

De ce ?

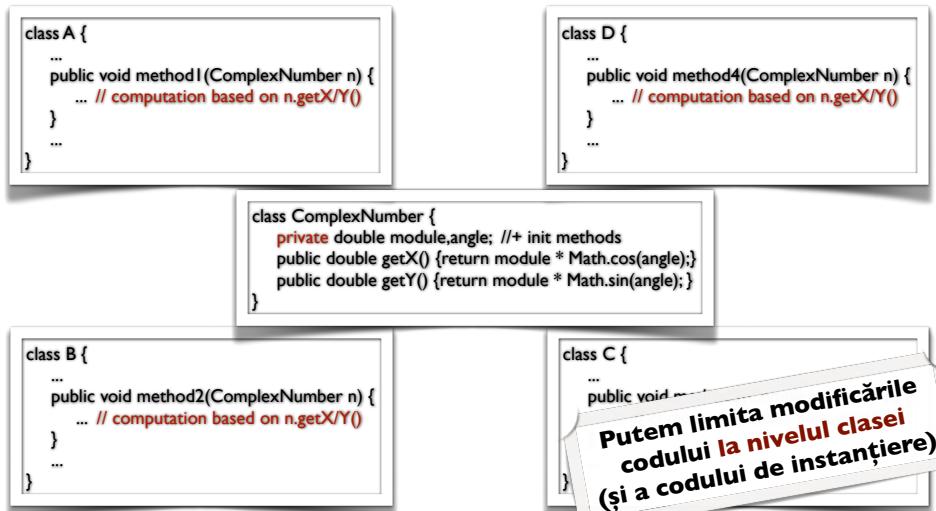


Dr. Petru Florin Mihancea

De ce ?



Dr. Petru Florin Mihancea



Dr. Petru Florin Mihancea

6

Membrii statici

Modifierul static

Dr. Petru Florin Mihancea

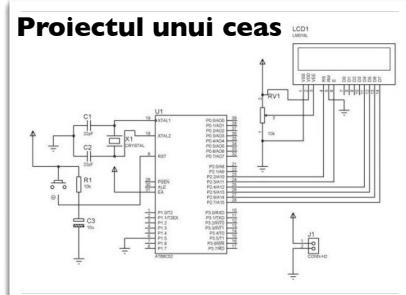


<http://www.timetools.co.uk>



Obiecte

NU. Dar a cui este ?



Dr. Petru Florin Mihancea

Clasa

<http://hubpages.com/hub/digital-clock-using-microcontroller-89C5289S52>

Întrebare ?



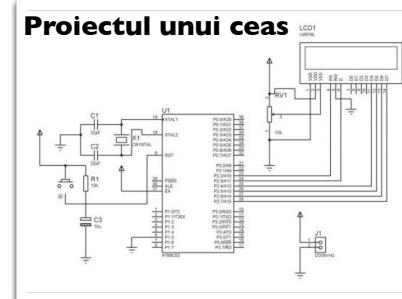
<http://www.timetools.co.uk>



Obiecte

Clasa

Vrem un contor care să numere câte obiecte ceas am creat. Este acesta o variabilă instată a unui obiect ceas ?



Dr. Petru Florin Mihancea

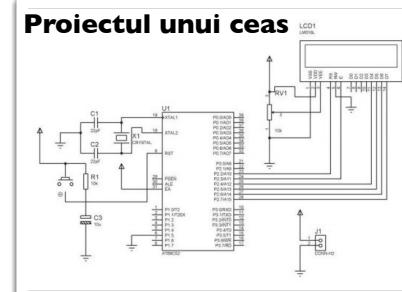


<http://www.timetools.co.uk>



Obiecte

Clasa



Dr. Petru Florin Mihancea

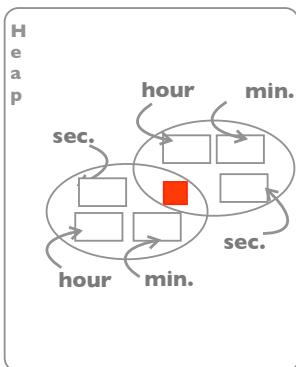
Este o variabilă a clasei !

Întrebare ?

Modificatorul static

```
class Clock {  
    ...  
    //numără câte ceasuri am creat  
    //variabilă/câmp/metodă de clasă  
    private static int count;  
    public Clock(int h, int m, int s) {  
        count++;  
        setTime(h, m, s);  
    }  
    public static int getCount() {  
        return count;  
    }  
}
```

toate obiecte au un singur exemplar



Dr. Petru Florin Mihancea

Modificatorul static (II)

metodele statice **NU** se execută pe un obiect al clasei

- **this** nu are sens în metodele statice
- nu poți accesa membrii instantă via **this** (implicit ori explicit)
- utilizare **NumeClasă.numeMetodă(params)**
ex. **Clock.getCount()**

```
class Clock {  
    private int hour, minute, seconds;  
    private static int count;  
    public static int oMetodaStatica(Clock c) {  
        c.hour = 0;  
        hour = 0; //Eroare de compilare  
        this.hour = 0; //Eroare de compilare  
    }  
    ...  
}
```

Dr. Petru Florin Mihancea

7

Variabile final

Modificatorul final pt. variabile

Dr. Petru Florin Mihancea

final

O astfel de variabilă poate fi atribuită o singură dată (altfel e eroare de compilare)

Un câmp instantă final trebuie să fie sigur inițializat până la sfârșitul oricărui constructor (altfel eroare de compilare)

```
class CatalogNote {  
    //Exemplu de utilizare: definirea de constante simbolice  
    //Prin convenție, numele lor se scrie nu litere mari  
    public final static int NOTA_MAXIMA = 10;  
}
```

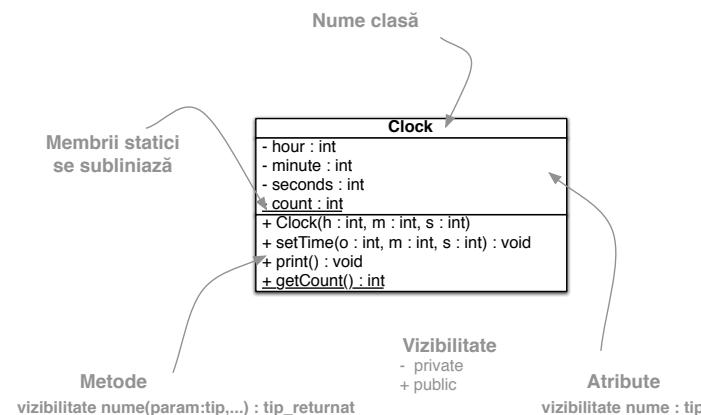
Dr. Petru Florin Mihancea

8

Unified Modelling Language Diagrama de clase sau Să vorbim **grafic** despre clase

Dr. Petru Florin Mihancea

Reprezentarea claselor



Dr. Petru Florin Mihancea

```
class Clock {
```

```
    private int hour, minute, seconds;  
    private static int count = 0;
```

```
    public Clock(int h, int m, int s) {  
        count++;  
        setTime(h, m, s);  
    }
```

```
    public void setTime(int h, int m, int s) {  
        hour = (h >= 0) && (h < 24) ? h : 0;  
        minute = (m >= 0) && (m < 60) ? m : 0;  
        seconds = (s >= 0) && (s < 60) ? s : 0;
```

```
}
```

```
    public void print() {  
        System.out.println("Current time " + hour + ":" + minute + ":" + seconds);  
    }
```

```
    public static int getCount() {  
        return count;  
    }
```

```
}
```

Dr. Petru Florin Mihancea

9

Noțiuni teoretice

Dr. Petru Florin Mihancea

Identitatea obiectului

... este acea proprietate a unui obiect care îl distinge de oricare alt obiect

referință / adresa de memorie unde e alocat

NU confundați numele de variabile referință cu identitatea unui obiect !

Bloch - OO Analysis and Design

```
class Main {  
    public static void main(String argv[]) {  
        ...  
        Clock firstClock, secondClock;  
        firstClock = new Clock(0,0,0);  
        firstClock . setTime(5, 10, 0);  
        secondClock = new Clock(0,0,0);  
        secondClock . setTime(19, 30, 45);  
        firstClock = secondClock;  
        firstClock . setTime(0, 0, 0);  
        secondClock.print();  
    }  
}
```

Dr. Petru Florin Mihancea

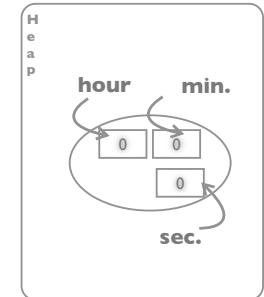
Starea obiectului

... reprezintă toate proprietățile obiectului plus valorile curente pentru fiecare din aceste proprietăți

Bloch - OO Analysis and Design

```
class Main {  
    public static void main(String argv[]) {  
        ...  
        Clock firstClock;  
        firstClock = new Clock(0, 0, 0);  
        //Momentul A
```

Starea la momentul A



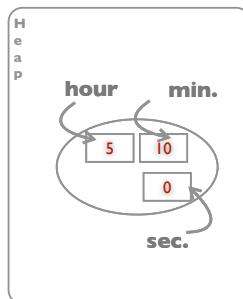
Dr. Petru Florin Mihancea

Starea obiectului

... reprezintă toate proprietățile obiectului plus valorile curente pentru fiecare din aceste proprietăți

```
class Main {  
    public static void main(String argv[]) {  
        ...  
        Clock firstClock;  
        firstClock = new Clock(0, 0, 0);  
        //Momentul A  
        ...  
        firstClock . setTime(5, 10, 0);  
        //Momentul B
```

Starea la momentul B



Dr. Petru Florin Mihancea

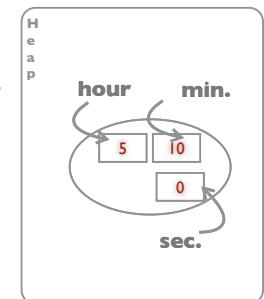
Starea obiectului

... reprezintă toate proprietățile obiectului plus valorile curente pentru fiecare din aceste proprietăți

Bloch - OO Analysis and Design

```
class Main {  
    public static void main(String argv[]) {  
        ...  
        Clock firstClock;  
        firstClock = new Clock(0, 0, 0);  
        //Momentul A  
        ...  
        firstClock . setTime(5, 10, 0);  
        //Momentul B  
        ...  
        firstClock.print(); //Current time 5:10:0
```

Starea la momentul B+



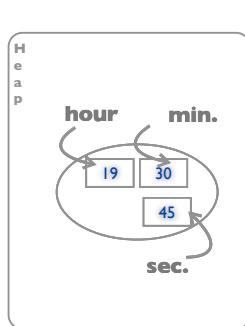
Starea se conservă !

Starea obiectului

... reprezintă toate proprietăile obiectului plus valorile curente pentru fiecare din aceste proprietăți

```
class Main {  
    public static void main(String argv[]) {  
        ...  
        Clock firstClock;  
        firstClock = new Clock(0, 0, 0);  
        //Momentul A  
        ...  
        firstClock.setTime(5, 10, 0);  
        //Momentul B  
        ...  
        firstClock.print(); //Current time 5:10:0  
        ...  
        firstClock.setTime(19, 30, 45);  
        //Momentul C  
        ...  
    }  
}
```

Starea la momentul C



Booch - OO Analysis and Design

Dr. Petru Florin Mihancea

Comportamentul obiectului

... reprezintă cum anume acționează/reacționează acel obiect în termeni de schimbare a stării sale și de interacțiune cu alte obiecte

Booch - OO Analysis and Design

setTime - ora/min/sec se schimbă la valoarea indicată sau 0 dacă valoarea dată e incorrectă

print - tipărește pe ecran ora indicată

poate apărea schimbare de stare în anumite condiții

ce se tipărește depinde de starea curentă

Comportamentul obiectului (II)

În esență, comportamentul unui obiect depinde de starea curentă cât și de operația efectuată pe obiect, iar unele operații pot altera starea sa



http://www.e-automaticeafera.ro/

Exemplu

Cum se comportă acest obiect ?

Interfață

... reprezintă setul de (declarații de) operații (de obicei publice) pe care un client le poate efectua pe un obiect

```
class Clock {  
    private int hour, minute, seconds;  
    public void setTime(int h, int m, int s) {  
        hour = (h >= 0) && (h < 24) ? h : 0;  
        minute = (m >= 0) && (m < 60) ? m : 0;  
        seconds = (s >= 0) && (s < 60) ? s : 0;  
    }  
    public void print() {  
        System.out.println("Current time " + hour + ":" + minute + ":" + seconds);  
    }  
}
```

restul ține de implementarea obiectului/clasei

Dr. Petru Florin Mihancea

Încapsularea

... compartimentarea elementelor unui obiect care formează structura și comportamentul său; încapsularea servește la separarea interfeței de implementarea abstractiunii

Bloch - OO Analysis and Design

Dr. Petru Florin Mihănea

Încapsularea (II)

... împachetarea datelor și a metodelor [ce lucrează cu acele date] în clase în combinație cu ascunderea implementării

Eckel - Thinking in Java

```
class Clock {  
    private int hour, minute, seconds;  
    public void setTime(int h, int m, int s) {  
        hour = (h >= 0) && (h < 24) ? h : 0;  
        minute = (m >= 0) && (m < 60) ? m : 0;  
        seconds = (s >= 0) && (s < 60) ? s : 0;  
    }  
    public void print() {  
        System.out.println("Current time " + hour + ":" + minute + ":" + seconds);  
    }  
}
```

Dr. Petru Florin Mihănea

Încapsularea (II)

... împachetarea datelor și a metodelor [ce lucrează cu acele date] în clase în combinație cu ascunderea implementării

Eckel - Thinking in Java

```
class Clock {  
    int hour, minute, seconds;  
    void setTime(int h, int m, int s) {  
        hour = (h >= 0) && (h < 24) ? h : 0;  
        minute = (m >= 0) && (m < 60) ? m : 0;  
        seconds = (s >= 0) && (s < 60) ? s : 0;  
    }  
    void print() {  
        System.out.println("Current time " + hour + ":" + minute + ":" + seconds);  
    }  
}
```

Dr. Petru Florin Mihănea

Încapsularea (II)

... împachetarea datelor și a metodelor [ce lucrează cu acele date] în clase în combinație cu ascunderea implementării

Eckel - Thinking in Java

```
class Clock {  
    ██████████ ██████████  
    public void setTime(int h, int m, int s) {  
        ██████████ ██████████  
    }  
    public void print() {  
        ██████████ ██████████  
    }  
}
```

Dr. Petru Florin Mihănea

Încapsularea (II)

... împachetarea datelor și a metodelor [ce lucrează cu acele date] în clase în combinație, cu ascunderea implementării

Încapsularea
(variație după autori)

Information/data hiding

Eckel - Thinking in Java

```
class Clock {  
    public void setTime(int h, int m, int s) {  
        ...  
    }  
  
    public void print() {  
        ...  
    }  
}
```

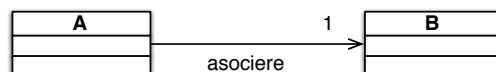
Dr. Petru Florin Mihăncea

Reprezentarea în diagrame UML de clase

```
class A {  
    private B b;  
    ...  
}
```

Multiplicitate
arată câte obiecte B are un A

alte variante uzuale
0..1 - zero sau cel mult unul
0..* - zero sau oricără de multe



asociere

agregare: un întreg (A) este compus din părți (B)
compoziție (ca noțiune UML) : un A are B-uri (pe vecie) numai și numai a lui
(a nu se confunda cu termenul object composition / compunerea)

Dr. Petru Florin Mihăncea

Componerea obiectelor

În esență, amplasarea de referințe la obiecte ca variabile instanță într-o clasă

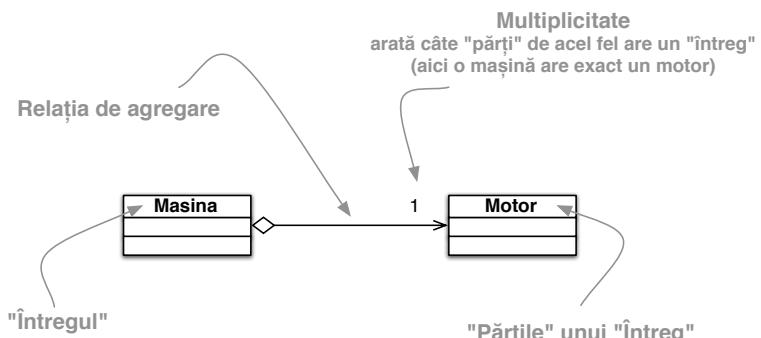
(inclusiv când referințele sunt într-un tablou variabilă instanță)

```
class Car {  
    private Engine engine;  
    public Car(Engine e) {  
        engine = e;  
    }  
    public void accelerate() {  
        engine.increaseFuelFlow();  
    }  
    ...  
}
```

```
class Engine {  
    public void increaseFuelFlow() {  
        ...  
    }  
}
```

Uzual spune că o mașină are un motor.
Componerea este o relație de tip has-a

Reprezentarea în diagrame UML de clase (II)



Dr. Petru Florin Mihăncea

10

Supraîncărcarea metodelor Overloading

Dr. Petru Florin Mihancea

```
class Clock {  
    ...  
    public void setTime(int h, int m) {  
        hour = (h >= 0) && (h < 24) ? h : 0;  
        minute = (m >= 0) && (m < 60) ? m : 0;  
        seconds = 0;  
    }  
    public void setTime(int h, int m, int s) {  
        hour = (h >= 0) && (h < 24) ? h : 0;  
        minute = (m >= 0) && (m < 60) ? m : 0;  
        seconds = (s >= 0) && (s < 60) ? s : 0;  
    }  
    ...  
}
```

Metode cu același
nume dar
semnături diferite

Dr. Petru Florin Mihancea

Semnătura unei metode

```
class Clock {  
    ...  
    //Modificatori tip_returnat nume(tip param1, ..., tip paramN) ...  
    public void setTime(int h, int m, int s) {  
        ...  
    }  
    ...  
}
```

Semnătura unei metodei

- **nume**
- **ordinea și tipurile parametrilor formali**
(implicit și numărul parametrilor :)

Dr. Petru Florin Mihancea

Supraîncărcarea (overloading)

Exemplu

Clasa referinței
System.out

```
PrintStream  
...  
+print(b : boolean) : void  
+print(c : char) : void  
+print(i : int) : void  
+print(s : double) : void  
+print(s : String) : void  
...  
+print(b : boolean) : void  
+println(b : boolean) : void  
+println(c : char) : void  
+println(i : int) : void  
+println(s : double) : void  
+println(s : String) : void  
...
```

```
int a = 1;  
String b = "something";  
System.out.println(a);  
System.out.println(b);
```

Dr. Petru Florin Mihancea

Merge și la constructori :)

```
class Clock {  
    ...  
    public Clock() {  
        count++;  
        hour = 12;  
        minute = seconds = 0;  
    }  
    public Clock(int h, int m, int s) {  
        count++;  
        setTime(h, m, s);  
    }  
    ...  
}
```

```
Clock a = new Clock();  
Clock b = new Clock(10,20,30);  
a.print();  
b.print();
```

OUTPUT
Current time 12:0:0
Current time 10:20:30

Dr. Petru Florin Mihăncea

11

Transmiterea parametrilor

Dr. Petru Florin Mihăncea

Apeluri la constructori

```
class Clock {  
    ...  
    public Clock() {  
        this(12,0,0);  
    }  
    public Clock(int h, int m, int s) {  
        count++;  
        setTime(h, m, s);  
    }  
    ...  
}
```

numai din alți constructori

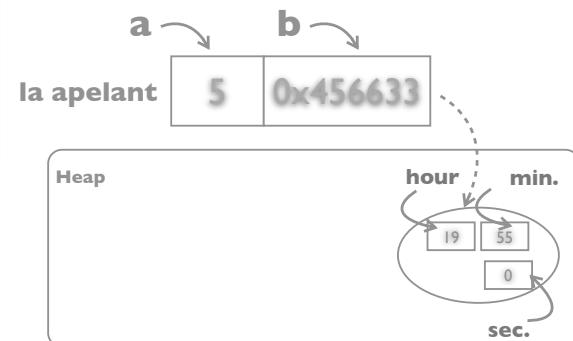
dacă e necesară această apelare, trebuie să fie prima instrucție

Dr. Petru Florin Mihăncea

Transmiterea parametrilor Java

prin valoare

```
void caller() {  
    int a = 5;  
    Clock b = new Clock();  
    b.setTime(19,55,0);  
    someobj.called(a, b);  
}
```



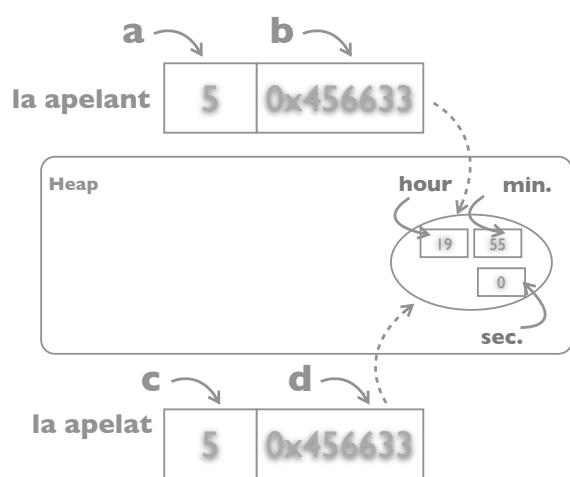
Dr. Petru Florin Mihăncea

Transmiterea parametrilor Java

prin valoare

```
void caller() {  
    int a = 5;  
    Clock b = new Clock();  
    b.setTime(19,55,0);  
    someobj.called(a, b);  
}
```

```
void called(int c, Clock d) {  
    //  
}
```

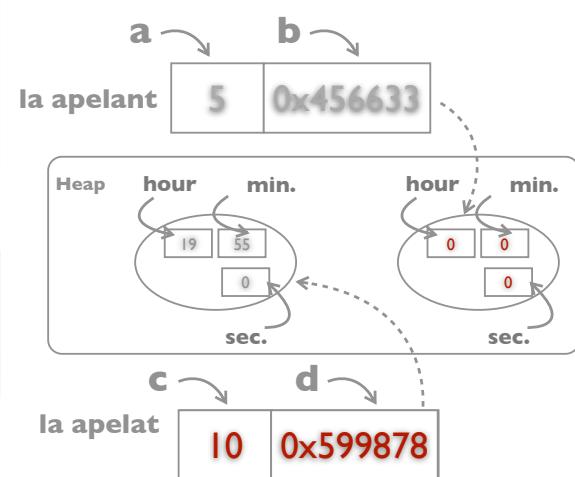


Transmiterea parametrilor Java

prin valoare

```
void caller() {  
    int a = 5;  
    Clock b = new Clock();  
    b.setTime(19,55,0);  
    someobj.called(a, b);  
}
```

```
void called(int c, Clock d) {  
    c = 10;  
    d.setTime(0,0,0);  
    d = new Clock(19,55,0);  
}
```

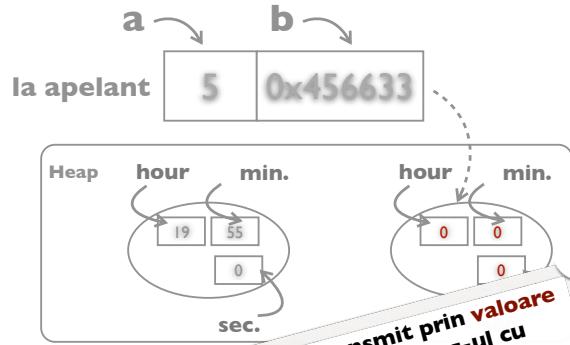


Transmiterea parametrilor Java

prin valoare

```
void caller() {  
    int a = 5;  
    Clock b = new Clock();  
    b.setTime(19,55,0);  
    someobj.called(a, b);  
    //a este tot 5 !  
    //b refera același obiect ceas (!)  
    //dar ora a fost schimbată la 0:0:0  
}
```

```
void called(int c, Clock d) {  
    c = 10;  
    d.setTime(0,0,0);  
    d = new Clock(19,55,0);  
}
```



12

Atenție la ...

Atenție la ce referă o referință !

```
class Main {  
    public static void main(String argv[]) {  
        Clock firstClock;  
        firstClock = null;  
        firstClock.setTime(0,0,0);  
        firstClock.print();  
    }  
}
```

OUTPUT

```
Exception in thread "main" java.lang.NullPointerException  
at nullpointer.Main.main(Main.java:7)
```

Dr. Petru Florin Mihăescu

Nume identice

```
class Clock {  
    private int hour, minute, seconds;  
    ...  
    public void setTime(int hour, int minute, int seconds) {  
        hour = (hour >= 0) && (hour < 24) ? hour : 0;  
        minute = (minute >= 0) && ( minute < 60) ? minute : 0;  
        seconds = (seconds >= 0) && ( seconds < 60) ? seconds : 0;  
    }  
    ...  
}
```



```
class Clock {  
    private int hour, minute, seconds;  
    ...  
    public void setTime(int hour, int minute, int seconds) {  
        this.hour = (hour >= 0) && (hour < 24) ? hour : 0;  
        this.minute = (minute >= 0) && ( minute < 60) ? minute : 0;  
        this.seconds = (seconds >= 0) && ( seconds < 60) ? seconds : 0;  
    }  
    ...  
}
```

Dr. Petru Florin Mihăescu