

Curs II

1. Note . Notare + examen

■ Detalii despre examen și despre calculul notei finale:

- Subiectele de examen vor fi de una sau mai multe forme din cele enumerate mai jos:
 - Întrebări cu răspunsuri scurte sub forma unui număr ori cuvânt (ex. răspuns determinat pe baza înțelegerii execuției unei secvențe de cod).
 - Întrebări "multiple choice multiple answers" ce permit selectarea uneia sau a mai multor răspunsuri considerate corecte dintr-o listă dată. Menționăm că i) selectarea unui răspuns greșit conduce la anularea întregului punctaj pentru întrebarea respectivă și ii) se va face punctare parțială dacă s-au selectat doar răspunsuri corecte însă nu toate răspunsurile corecte din listă.
 - Întrebări ce solicită discuția unei probleme / argumentarea validității sau invalidității unei afirmații sub forma unui ese (ex. descrierea unui concept teoretic)
 - Solicitarea scrierii unei secvențe de cod (ex. corectarea unei secvențe de cod pentru a respecta un principiu de POO).
 - Orice combinație a alternativelor anterioare
- Număr de întrebări și timp de lucru - TBD
- Notă finală a disciplinei:
 - K1 (Examen) 50%
 - K2 (Activitate pe parcurs) 50%
- C. Marinescu, P.F. Mihancea, Programare Orientată pe Obiecte în Limbajul Java, este disponibilă aici [aici](#).
- tar.gz este un soi de zip (ar trebui să meargă cu programe de dezarchivare pt. zip)
- În partea de jos de la primul slide este un număr de versiune. Verificați să fiți up-to-date.

1. Accesul la membrii unei clase

Se face cu ajutorul modificatorilor / specificatorilor de acces

private → pot fi accesat doar în interiorul clasei

public → pot fi accesat de oriunde

```
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
    }
}
```

→ sunt în interiorul clasei

este referință în clasa

→ nu sunt în interiorul clasei

Se ce trebuie să pun clasele pe privat?

- Toate clasele nefolosite în altă clare se pun pe private
- Punem clasele pe privat și facem funcții publice care să returneze clasele
- În OOP nu e bine să dau acces la clasele folosite, sau acces la valori prin juncțiile publice care returnează valori.

2. Membri statici

Membrii statici nu se amâna cu cel din C.

Puteam considera că obiecte am creat pentru un câmp static.

Când folosim un câmp static el nu sunt pus în stoc ca element

Metodele statici nu se execută pe un obiect al clasei:

- this nu are sens în metodele statică
- Nu pot accesa membru instanță via this (implicit/explicit)

this. Ceea ce → eroare de compilare

De ce funcția main trebuie să fie publică și statică?

Dacă ar fi privată compilatorul nu ar avea acces la main

Dacă ar fi non-statică compilatorul nu ar găsi obiectul main.

Dacă definesc static o saloare o am executat o singură dată. În cîmpul de memorie, practic declarăm 10 obiect, și variabila de tip static ocupă un singur loc în memorie.

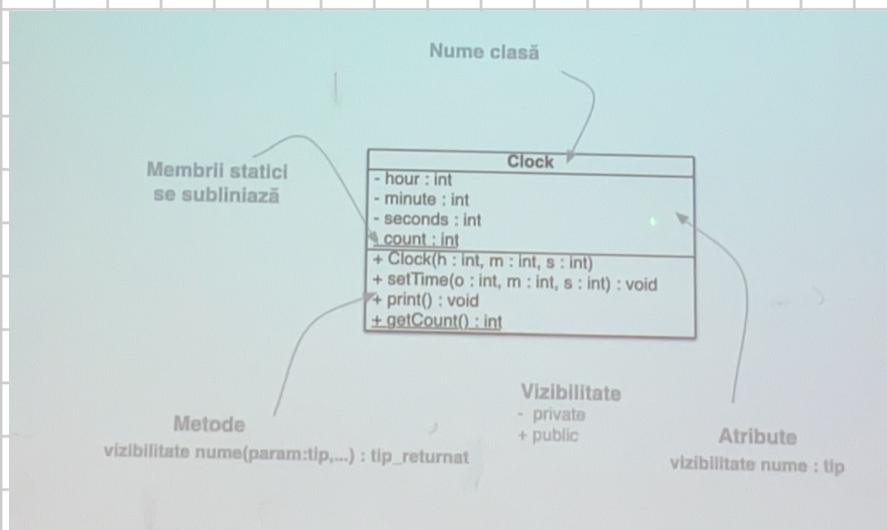
3. Variabile finale

Poate fi initializată (atribuită) o singură dată.

Trebuie initializată pînă la finalul constructorului:

public final static int Nota_Maxima = 10;

4. Diagrama de clase (U.M.L.)



5. Noțiuni teoretice

• Identitatea obiectului este proprietatea de a distinge un obiect de alt obiect.

Nu confundați numele de variabilă referintă cu identitatea unui obiect!

• Starea obiectului reprezintă toate proprietățile obiectului plus valoarele curente pentru fiecare din aceste proprietăți.

Starea se conservă! Variabilele statice nu fac parte din starea unui obiect.

- Comportamentul obiectului reprezintă cum anume acționează acel obiect în termeni de schimbare a sărării sale și de interacțiune cu alte obiecte.
- Interfața reprezintă setul de operații pe care un client le poate efectua pe un obiect.

- ⁴Încapsularea împachetăaza datele și a metodelor în clase în combinație cu ascunzarea implementării.

- Componerea obiectelor este în esență, amplasarea de referințe la obiect ca variabile instanță într-o clasă.

Componerea este o relație de tip has-a.

6. Supraîncărcarea obiectelor

- Semnatura unei metode:

- nume

- ordinea și tipul parametrilor formalii

Supraîncărcarea este permisă în Java.

Metode cu același nume dar semnături diferite.

Clasa referinței (System.out) este un exemplu de supraîncărcare. Practic se poate aplica orice tip de date.

int a = 1;

String b = "orice";

System.out.println(a); } System.out.println(int...)
System.out.println(b); } =) System.out.println(char...)

Supraîncărcarea merge și la constructor.

class Clock {

 public Clock () { ... }

 public Clock (int ora, int minut, int secunda) { ... }

}

Apeluri la constructori munca din alti constructori.

public Clock () {

 this(12, 0, 0);

}

7. Transmiterea parametrilor

Dacă transmitem o valoare ca nu se schimbă, în schimb dacă transmit un obiect acesta își schimbă valoarea din cauza acției de memorie.

8. Atenție la ce referești o referință

Un pointer se poate defini null.

Numele identice ale variabilelor nu sunt acceptate.

class Clock {

private int ora, minut;

public int Set(int ora, int minut)

}

ora = 10;

minut = 23;

}

}

Eroare de compilare

class Clock {

private int ora, minut;

public int Set(int ora, int minut)

}

this.ora = 10;

this.minut = 23;

}

}

Corect