Laborator 2

Calculul Salariilor

Fie o firmă care are mai multe categorii de angajați.

Pentru aplicația pe care o veți realiza va lua în considerare doar două categorii: **ingineri** și **șoferi**.

In firmă este fixat un salariu orar minim de 15 lei/oră, iar fiecare categorie are un **coeficientSlarial** propriu. Salariul orar al unei categorii de angajati este egal cu

salariulOrarCategorie = salariuOrarMinim x coeficientSlarial_categorie

Inginerii sunt plătiți proporțional cu nr de ore lucrate

salariuInginer = salariulOrarInginer x nrOreLucrate

Pentru un inginer **coeficientSlarial** = 1.5, iar numărul de ore lucrate pe luna nu poate fi mai mare de 250.

Șoferii sunt plătiți proporțional cu nr de ore lucrate la care se adaugă un bonus care depinde de kilometrii parcurși

salariuSofer = salariulOrarSofer x nrOreLucrate + kmParcursi x 0.1

Pentru toți șoferii avem **coeficientSlarial** = 1.00, iar numărul de ore lucrate pe lună nu poate fi mai mare de 300 și km parcurși nu pot fi mai mulți de 5000.

Interfata IAngajat

Această interfață va fi implementată în mod obligatoriu de clasele **Inginer** și **Șofer**.

```
public interface IAngajat {
    public static final double salariuOrarMinim=15;
    public void setNrOreLucrate(int nrOreLucrate);
    public double salariu();
}
```

Nota. In interfetele Java variabilele sunt **implicit** `public static final` deoarece

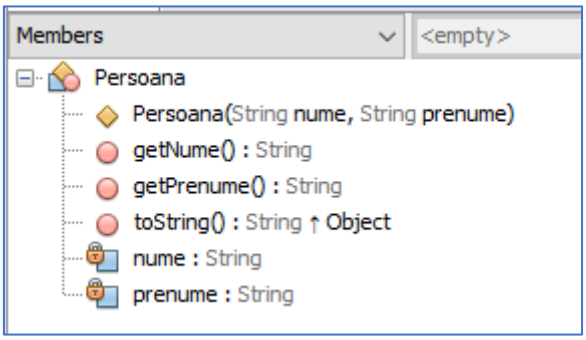
- `public` – trebuie sa fie vizibile in toate clasele care implementeaza interfata
- `static` – trebuie sa existe inainte de orice instanta a unei clase care o implementeaza
- `final` - fiind comune tuturor claselor care implementeaza interfata ele nu trebuie sa poata fi modificate

De aceea prezenta in declaratie a `public static final este` este redundanta (declaratia putea fi doar `double salariuOrarMinim=15;`)

1. Creati clasele Persoana, Inginer, Sofer, CalculSalarii (5p)

- **Persoana** (nume, prenume)

Aceasta clasa va avea doar câmpurile și metodele din figura următoare:



- **Inginer** – extinde **Persoana** și implementeaza **IAngajat**

- **Sofer** – extinde **Persoana** și implementeaza **IAngajat**

Clasele **Inginer** și **Sofer** :

- vor avea toate campurile necesare pentru a produce afisarea ceruta de aplicatie
- ***IMPORTANT:*** metodele toString() din **Inginer** și **Sofer** vor folosi metoda toString() din clasa **Persoana**.

- **CalculSalarii** – va fi o clasa care contine doar metoda **main()** în care
  - se instantiaza 3 obiecte: 2 ingineri și un șofer (folositi datele din exemplul de iesire de mai jos)
  - se actualizeaza numarul de ore lucrate la 100, 200 și respectiv 250 de ore,
  - se actualizeaza și faptul ca șoferul a parcurs 1000 km
  - se afiseaza salariul fiecaruia
  - se afiseaza ***Total salarii*** calculat ca suma salariilor angajatilor

Iesirea produsa de program **trebuie** sa fie ca cea de mai jos

```
Inginer Barbulescu Barbu a lucrat 100 ore - slariu=2250.0
Inginer Trestie Tudor a lucrat 200 ore - slariu=4500.0
Sofer Repede Raul a lucrat 250 ore, a parcurs 1000 km - slariu=3850.0
Total salarii=10600.0
```

Referitor la punctajul acordat:

- daca metodele toString() din **Inginer** și **Sofer** NU vor folosi metoda toString() din clasa **Persoana** atunci punctajul se va diminua cu 2x 0.5p = 1p
- daca – clasele Inginer și Sofer NU vor extinde **Persoana** și NU implementeaza **IAngajat** atunci punctajul se va diminua cu 2 x (2x 0.5p) = 2p

2. Rescrieti aplicatia renuntand la interfata IAngajat

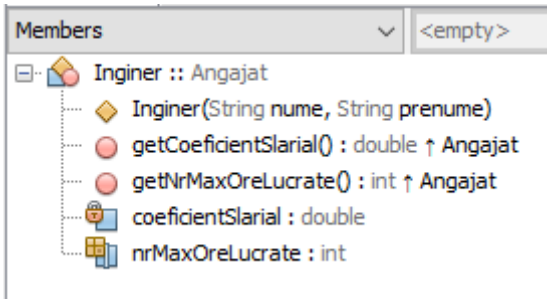
(3p)

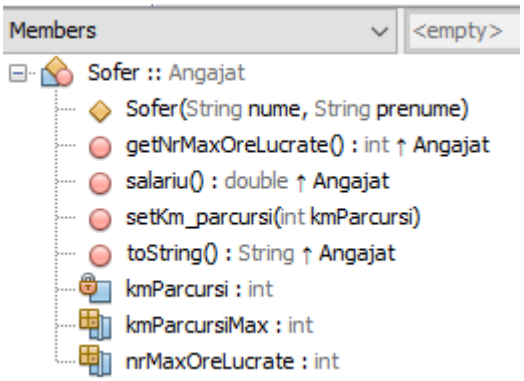
Faceti o copie a proiectului precedent, redenumiti copia salariiAngajati și faceti modificarie în aceasta copie astfel incat la sfarsitul laboratorului sa puteti prezenta ambele proiecte pentru acordarea de puncte.

Indicatie: Veti scrie o clasa **Angajat** care sa extinda clasa **Persoana**. Clasele **Inginer** și **Sofer** vor extinde **Angajat**.

Se acorda 1 p dacă se produce aceeași ieșire cu aceleași date de intrare.

Celelalte 2p se acordă dacă clasele **Inginer** și **Sofer** **au cel mult aceeași membri** – metode și campuri - ca în figura de mai jos.





Variabila mrMaxOreLucrate are valoarea 250 pentru orice inginer și 300 pentru orice Șofer, iar variabila kmParcursiMax are valoarea 5000 pentru orice șofer.

3. Scrieți clasa Firma (2p)

Clasa **Firma** va memora toți angajații și va permite preluarea orelor lucrate de toti angajatii dintr-un fișier **pontaj.txt** care va avea formatul ca în exemplul urmator

```
101 100
23 200
302 250 1000
```

Primul numar este ID-ul propriu fiecarui angajat. Un intreg. Nu exista 2 angajati diferiti cu acelasi ID. Veti completa în mod corespunzator aplicatia din ultimul proiect.
Dupa ID, celalalt numar/celelalte numere de pe o linie din fisier reprezinta numarul de ore lucrate, eventual numarul de km parcursi.

Clasa **Firma** va avea și metode care:

- Afiseaza salariile tuturor angajatilor
- Calculeaza ***Total salarii***

Veti scrie si o metoda

```
Angajat getAngajatDupaID (int ID);
```

care va returna angajatul cu ID-ul primit ca argument.

Link-uri utile

1. Pentru documentatia claselor Java consultati https://docs.oracle.com/javase/8/docs/api/
2. O aplicatie asemanatoare este prezentata în cartea recomandata: *Building Java Programs: A Back to Basics Approach, 5th edition, by Stuart Reges and Marty Stepp*, Consultati prezentarea pt. cap. 9 de la adresa http://www.buildingjavaprograms.com/supplements5.shtml
3. Daca utilizati clasa **Scanner** atentie dacă alternati nextInt() cu nextLine(). A se vedea prezentarea din cap.6 din aceeasi pagina web http://www.buildingjavaprograms.com/supplements5.shtml