Programarea orientată pe obiecte

SERIA CB

CLASE INTERNE SIMPLE

O clasă internă este o clasă **definită în interiorul altei clase**. În exemplul de mai jos este definită clasa `Motor` în interiorul clasei `Masina`.

```
public class Masina {
    private int kmParcursi;

    public void afiseazaInBord() {
        System.out.println("Km parcursi: " + kmParcursi);
    }

    class Motor {
        public void accelereaza() {
            kmParcursi++;
            afiseazaInBord();
        }
    }
}
```

- Fișierele compilate se vor numi Masina.class și Masina$Motor.class;
- Clasa internă poate accesa variabilele membru și metodele clasei în care este definită deoarece **clasa internă este în sine un membru al clasei externe**;
- În practică, obiectele din clasa internă sunt în multe cazuri create în clasa externă. Dar se pot crea obiecte din clasa internă și în alte clase.
- Dacă clasa internă este non-statică (nu are cuvântul static în declarație), **trebuie întâi creată o instanță a clasei externe**, apoi folosită următoarea sintaxă pentru a crea un obiect al clasei interne:

```
public static void main(String[] args) {
    Masina masina = new Masina();
    Masina.Motor motor = masina.new Motor();
    motor.accelereaza();
}
```

sau

```
public static void main(String[] args) {
    Masina.Motor motor = new Masina().new Motor();
    motor. accelereaza();
}
```

- Dacă clasa internă este **statică**, atunci se folosește următoarea sintaxă:

```
Masina.Motor motor = new Masina.Motor();
```

- Referința `this` în cadrul clasei interne se referă la clasa internă, iar pentru clasa externă se folosește `Extern.this`:

```
class Extern {
    private int x = 5;

    public void creeazaIntern() {
        Intern intern = new Intern();
        intern.afiseazaExtern();
    }

    class Intern {
        public void afiseazaExtern () {
            System.out.println("x =" + x);
            System.out.println("Intern = " + this);
            System.out.println("Extern = "+ Extern.this);
        }
}

public static void main(String[] args) {
    Extern.Intern intern = new Extern().new Intern();
    intern.afiseazaExtern();
    }
}
```

În urma rulării metodei `main` se va afișa:

```
x = 5
Intern = Extern$Intern@113708
Extern = Extern@33f1d7
```

- Modificatorii de acces care pot fi aplicați claselor interne sunt: `final`, `abstract`, `public`, `private`, `protected`, `static`.

CLASE INTERNE STATICE

O **clasă internă statică (sau nested)** este o clasă care **este un membru static al clasei externe**. Modificatorul `static` indică faptul că această clasă este un membru static al clasei externe, ceea ce înseamnă că *poate fi accesat direct, fără a avea nevoie de o instanță a clasei externe*.

```
class Extern {
    static class A {
        void metoda() {
            System.out.println("Apel metoda");
        }
    }
}
class Main {
    public static void main(String[] args) {
        Extern.A a = new Extern.A();
        a.metoda();
    }
}
```

```java
class Extern {
    public int membruExtern = 10;
    class A {
        private int i = 1;
        public int metoda() {
            return i + Extern.this.membruExtern;
        //Se poate accesa un membru al clasei externe
        }
    }

    static class B {
        public int j = 2;
        public int metoda() {
            return j + membruExtern;
            //EROARE, nu putem accesa un membru ne-static al clasei externe
        }
    }
}

public class Test {
    public static void main(String[] args) {
        Extern ext = new Extern();
        Extern.A a = ext.new A();
        //instantiere corecta pentru o clasa ne-statica
        Extern.B b1 = ext.new B();
        //instantiere incorectă a clasei statice
        Extern.B b2 = new Extern.B();
        //instantiere corecta a clasei statice
    }
}
```

Așa cum o metodă statică nu are acces la variabilele membru și metodele non-statice ale clasei, ***o clasă statică nu are acces la variabilele membru și metodele non-statice ale clasei externe****!*

APLICAȚII:

Listă simplu înlănțuită

Implementați clasa `Lista`, ce conține o clasa internă `Nod`.

Clasa internă `Nod` are ca variabile membru:

```java
private int val;
private Nod urm;
```

`val` reprezintă valoarea stocată în nodul respectiv, iar `urm` reprezintă adresa către nodul următor. Se impune restricția ca `val >= 0`

Clasa `Nod` conține un constructor cu parametri și unul fără parametri:

```java
public Nod(int x) {
   val = x;
   urm = null;
}
public Nod(){
}
```

Clasa `Lista` are ca variabile membru:

```
private Nod varf;
private Nod coada;
private int contor;
```

Clasa `Lista` va avea următoarele metode:

- `public Lista()` – creează o nouă listă prin instanțierea obiectelor `varf` și `coada` și inițializarea lui `contor` cu valoarea `0`. La început, `varf = coada = null`.
- `public void adauga(int x)` – adaugă un nod nou cu valoarea `x` la finalul listei sau ca prim element, dacă lista este vidă
- `public int dimensiune()` – returnează numărul de elemente din listă (`contor`)
- `public void afisareLista()` – afișează elementele listei, de la vârf spre coadă
- În funcția `main` creați o listă ce va conține elemente mai mari sau egale cu `0`, afișați elementele ei, iar apoi apelați metodele definite în clasă.

Contor liste

Adăugați clasei `Lista` un o clasă internă statică `ContorListe`, ce va conține un câmp static `numarListe` și o metodă nestatică `incrementNumarListe()` care va incrementa acel contor – această metodă va fi apelată la crearea unei noi liste (în constructorul clasei `Lista`). Modificați metoda `afisareLista()` din clasa `Lista` pentru a afișa și numărul total de liste create.

Book

- ✓ Creați clasa `Book`, cu variabila membru `private String author`
- ✓ Definiți un constructor cu parametrul `author` și suprascrieți metoda:
  `public String toString()`

FirstPage

- ✓ Creați clasa nestatică `FirstPage`, ce conține variabila membru `private String poem`.
- ✓ Definiți un constructor cu parametrul `poem`, astfel:
  - dacă autorul este Eminescu, setați ca poezie:

    Un cer de stele dedesubt

    Deasupra-i cer de stele

    Parea un fulger ne'ntrerupt

    Ratacitor prin ele
  - dacă autorul este Alecsandri, setați ca poezie:

    In paduri trasnesc stejarii! E un ger amar, cumplit!

    Stelele par inghetate, cerul pare otelit

    Iar zapada cristalina pe campii stralucitoare

    Pare-un lan de diamanturi ce scartaie sub picioare

- dacă autorul este Cosbuc, setați ca poezie:

  Sunt copii. Cu multe sanii

  De pe coasta vin tipand

  Si se-mping si sar razand

  Prin zapada fac matanii

  Vrand-nevrand

✓ Definiți metoda `public int numberVowels()`, ce returnează câte vocale conține `poem`

✓ Definiți un getter pentru `poem`

✓ Actualizați metoda `public String toString()` din clasa `Book`, astfel încât să returneze autorul cărții, poezia de pe prima pagină și numărul ei de vocale

✓ În `main`, creați 3 cărți (cu autorii Eminescu, Alecsandri și Coșbuc) și afișați informațiile despre ele, ținând cont de implementarea din `toString()`.

Exemplu:

```
Book book1 = new Book("Eminescu");
System.out.println(book1);
```

Referințe:

1. Oana Balan, Mihai Dascalu. **Programarea orientata pe obiecte in Java**. Editura Politehnica Press, 2020
2. Bert Bates, Kathy Sierra. **Head First Java: Your Brain on Java - A Learner's Guide 1st Edition**. O'Reilly Media. 2003
3. Herbert Schildt. **Java: A Beginner's Guide**. McGraw Hill. 2018
4. Barry A. Burd. **Java For Dummies**. For Dummies. 2015
5. Joshua Bloch. **Effective Java**. ISBN-13978-0134685991. 2017
6. Harry (Author), Chris James (Editor). **Thinking in Java: Advanced Features (Core Series) Updated To Java** 8
7. Learn Java online. Disponibil la adresa: https://www.learnjavaonline.org/
8. Java Tutorial. Disponibil la adresa: https://www.w3schools.com/java/

   The Java Tutorials. Disponibil la adresa: https://docs.oracle.com/javase/tutorial/