

Laborator 6: Clase interne (partea I)

1 Responsabil

Gabriel Guțu-Robu, gabriel.gutu@upb.ro

Publicat: 13 noiembrie 2022, 10:12

2 Clase interne simple

O clasă internă este o clasă definită în interiorul altei clase. În exemplul de mai jos este definită clasa `Intern` în interiorul clasei `Extern`.

```
public class Masina {
    private int kmParcursi;

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

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

O clasă internă are următoarele proprietăți:

- 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@33fld7
```

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

3 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();
    }
}
```

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!

```
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
    }
}
```

4 Aplicații

4.1 Listă simplă înlănțuită

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

Clasa internă `Nod` are ca variabile membru:

```
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:

```
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`.
- `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
- `public void adaugaPozitie(int index, int x)` – adaugă un nod nou cu valoarea `x` pe poziția `index`. Se va verifica dacă poziția `index` este validă. Cazurile de excepție sunt atunci când `index < 0` sau `index > dimensiune()`. Atunci când `index == dimensiune()`, se va adăuga un nou nod în plus la finalul listei.
- `public int elementPozitie(int index)` – returnează valoarea nodului de pe poziția `index`. Se va verifica dacă poziția `index` este validă. Cazurile de excepție sunt atunci când `index < 0`, `index >= dimensiune()` sau `dimensiune() == 0`
- `public int elimina(int index)` – elimină nodul de pe poziția `index` și se returnează valoarea lui. Se va verifica dacă poziția `index` este validă. Cazurile de excepție sunt atunci când `index < 0`, `index >= dimensiune()` sau `dimensiune() == 0`
- `public void seteaza(int index, int x)` – setează pe poziția `index` valoarea `x`. Se va verifica dacă poziția `index` este validă. Cazurile de excepție sunt atunci când `index < 0`, `index >= dimensiune()` sau `dimensiune() == 0`
- `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 apălați metodele definite în clasă.

4.2 Contor liste

Adăugați clasei Lista un o clasă internă statică ContorListe, ce va conține un câmp static numarListe și o metodă 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.