METODE AVANSATE DE PROGRAMARE

Conf.univ.dr. Ana Cristina DĂSCĂLESCU

Temtică curs 5

- Principilu de desing pentru clasă imutabilă
- Record

CLASE IMUTABILE

- O **clasă este imutabilă** dacă nu mai putem modifica conținutul unei instanțe a sa (un obiect) după creare.

- Orice modificare a obiectului respectiv presupune crearea unui nou obiect și înlocuirea referinței sale cu referința noului obiect creat.

- În limbajul Java există mai multe clase imutabile predefinite: String, clasele înfășurătoare (Integer, Float, Boolean etc.), BigInteger etc.

CLASE IMUTABILE

- Principalele avantaje ale utilizării claselor imutabile sunt următoarele:
  - **sunt implicit thread-safe** (nu necesită sincronizare într-un mediu concurent);
  - sunt ușor de proiectat, implementat, utilizat și testat;
  - sunt mai rapide decât clasele mutabile;
  - pot fi utilizate sub forma unei chei în structuri de date asociative (de exemplu, tabele de dispersie - **`HashMap`**);

CLASE IMUTABILE

➢ De obicei, crearea unei clase imutabile trebuie să respecte următoarele reguli:

1. clasa nu va permite rescrierea metodelor sale, fie declarând **clasa de tip `final`**, fie declarând constructorii ca fiind `private` și folosind metode de tip `factory` pentru a crea obiecte;

2. toate câmpurile vor fi declarate ca fiind **`final`** (li se vor atribui valori o singură dată, printr-un constructor cu parametri) și **`private`** (nu li se pot modifica valorile direct);

3. clasa **nu va conține metode de tip `set`** sau alte metode care pot modifica valorile câmpurilor;

CLASE IMUTABILE

4. **dacă există câmpuri care sunt referințe spre obiecte mutabile**, se va împiedica modificarea acestora, astfel:

   a. nu se vor folosi referințe spre obiecte externe, ci spre copii ale lor (se va folosi compoziția, ci nu agregarea!)

   b. **nu se vor returna referințe spre câmpurile mutabile, ci se vor returna referințe spre copii ale lor:**

RECORDS

- În Java 15 au fost introduse ***clasele de tip înregistrare*** (***records***).

- **O *înregistrare*** este o **clasă imutabilă** utilizată pentru a manipula o mulțime fixă de valori, denumite *componentele înregistrării*.

- De obicei, înregistrările sunt utilizate pentru încărcarea unor date dintr-o anumită sursă (de exemplu, un fișier sau o bază de date) și, eventual, transportarea acestora către o anumită destinație, folosind facilitățile limbajului Java pentru programarea în rețea.

- O înregistrare se declară într-un mod foarte concis, precizând doar tipul și numele componentelor sale în descriptorul înregistrării:

```
[modificatori de acces] record Denumire (descriptor) {}
public record Student(String nume, int grupa, double medie) {}
```

RECORDS

- Orice înregistrare este în mod implicit o clasă de tip **`final`** care extinde clasa **`java.lang.Record`**, deci o înregistrare nu poate fi abstractă, nu poate fi extinsă și nici nu poate extinde alte clase sau alte înregistrări.

- Pentru o înregistrare, compilatorul va genera automat o clasă de tip final având următoarele componente:

✓ câte o dată membră **privată și finală** pentru fiecare componentă;
✓ un **constructor canonic public** care va avea câte un parametru pentru fiecare componentă a înregistrării
✓ câte o metodă de **tip `get`** pentru fiecare componentă
✓ implementarea metodei `equals(Object)` din clasa `Object`;
✓ implementarea metodei `hashCode()` din clasa `Object`;
✓ implementarea metodei `toString()`din clasa `Object`.

RECORDS

- În cadrul unei înregistrări se pot adăuga **constructori supraîncărcați**, dar aceștia trebuie să apeleze explicit constructorul canonic:

```
record Student(String nume, int grupa, double medie) {
    public Student(String nume, int grupa) {this(nume, grupa, 0);}}
```

- **Constructorul canonic poate fi redefinit**, de obicei pentru a realiza prelucrări suplimentare sau validări ale componentelor înregistrării:

```
record Student(String nume, int grupa, double medie) {
    Student(String nume, int grupa, double medie) {
        if(medie < 0)
            medie = -medie;
        this.nume = nume.toUpperCase();
        this.grupa = grupa;
        this.medie = medie;    }}
```

RECORDS

- O înregistrare este o clasă imutabilă de tip ***shallowly immutable***, respectiv datele membre de tip referință vor fi copiate superficial (*shallow copy)*

- Pentru a evita acest aspect, se redefinește constructorul canonic și metodele de tip get corespunzătoare componentelor mutabile conform regulii 4 prezentate în secțiunea dedicată claselor imutabile!!!!

- Într-o înregistrare **nu se pot declara date membre de instanță**, dar se pot declara metode nestatice. De asemenea, se pot adăuga date, metode și blocuri de inițializare statice