

Tutoriat 4

Overloading și overriding



1. Supraîncarcarea (overloading)

a. Cum se face?

Prin declararea a 2 sau mai multe metode, în interiorul aceleiași clase, care au **nume identic**, dar antet diferit (**tipul parametrilor diferă/numărul parametrilor diferă**).

Obs:

- *Tipul retransmis nu contează*
- *Supraîncarcarea funcțiilor se poate realiza și în afara claselor (independent de acestea).*

```
class A {
public:
    int f() {
        cout<<"Functia int f()"<<endl;
    }
    void f() {
        cout<<"Functia void f()"<<endl;
    }
};

int main() {
    A a;
    a.f(); // eroare
    return 0;
}
```

```
class A {
public:
    int f() {
        cout<<"Functia int f()"<<endl;
    }
    int f(int x) {
        cout<<"Functia int f(int x)"<<endl;
    }

    int f(double x) {
        cout<<"Functia int f(double x)"<<endl;
    }
}
```

```
};
int main() {
    A a;
    a.f(); // Functia int f()
    a.f(2); //Functia int f(int x)
    a.f(3.4); //Functia int f(double x)
    return 0;
}
```

b. Ascunderea metodelor (Hiding)

i. Ce face?

Anulează procesul de supraîncărcare.

ii. Când se realizează?

Doar la **moștenirea** claselor.

Obs: Folosim operatorul de rezoluție(::) alături de numele clasei de **bază**, dacă dorim apelul metodei din clasa de bază, făcut printr-o instanță a clasei derivate.

```
class Baza {
public:
    void f() {
        cout<<"f() din clasa de baza"<<endl;
    }
};
class Derivata : public Baza {
public:
    void f(int x) {
        cout<<"f(int x) din clasa derivata"<<endl;
    }
};
int main() {
    Derivata d;
    d.f(); //eroare
    d.Baza::f(); //ok -> afiseaza: f() din clasa de baza
    d.f(2); // ok -> afiseaza: f(int x) din clasa derivata

    Baza b;
    b.f(); //ok -> afiseaza: f() din clasa de baza

    return 0;
}
```

2. Suprascrierea (overriding)

a. Când?

Suprascrierea se realizează la **moștenirea** claselor.

b. Ce face?

Metoda suprascrisă este înlocuită complet de cea care suprascrie.

c. Cum?

Metodele clasei **au același nume și aceiași parametrii**.

Obs: La fel ca la supraîncărcare, folosim operatorul de rezoluție(::) alături de numele clasei de **bază**, dacă dorim apelul metodei din clasa de bază, făcut printr-o instanță a clasei derivate.

```
class Baza {
public:
    void f() { cout << "Metoda f() din clasa de baza"<<endl; }
};
class Derivata : public Baza {
public:
    void f() { cout << "Metoda f() din clasa derivata"<<endl; }
};
int main() {
    Derivata d;
    d.Baza::f(); // Metoda f() din clasa de baza
    d.f(); // Metoda f() din clasa derivata
    return 0;
}
```