Programare orientată obiect

Cursul 8

Sumar

- Moștenire/derivare (3)
  - Funcții virtuale pure
    - Clase abstracte
    - Interfețe
  - Moștenirea multiplă
    - Moștenire virtuală

Funcții virtuale pure

- Prototip
  - virtual tip numeFunctie(param) **= 0**;
- Clasa include doar declarațiile, fără corpul funcțiilor
  - Funcțiile nu sînt implementate
- Suport pentru definirea:
  - Claselor abstracte
  - Interfețelor (din alte limbaje)

Clase abstracte

- Conțin cel puțin o funcție virtuală pură
  - Pot implementa metode
- Nu permit instanțierea obiectelor
- Create pentru derivare
- Daca nu sînt implementate în clasele derivate acestea vor fi abstracte
- Un contract pe care se obligă să îl respecte clasele derivate

Clase abstracte

- Obiect grafic
- Convertor documente
- Colecție
- Serializator
- Export

Clase abstracte

```
class Figura
{
protected:
        char *descriere;
public:
        virtual void deseneaza () = 0;
        //…
};
```

```
class Linie : public Figura
{
        int x1,x2, y1,y2;
public:
        void deseneaza();
        //…
};
```

Clase abstracte

Nu

- Figura fig;
- Figura *pFig = new Figura;

Da

- Figura *pFig;
- Figura *pFig = new Linie;
- Linie lin;
- Figura &rFig = lin;

Interfețe în C++

- Nu sînt definite
- Pot fi simulate cu ajutorul claselor abstracte:
  - Toate metodele sînt funcții virtuale pure
  - Toate metodele sînt publice
  - Nu conțin variabile de instanță
  - Pot avea date membre constante
- Uzual, relație de tipul: POATE-FACE (CAN-DO)
- Adaugarea de facilități unor clase

Interfețe

- Comparabil
- Editabil
- Clonabil
- Enumerabil
- Măsurabil
- Imprimabil

Interfețe

```
class IImprimabil
{
public:
        virtual void imprima() = 0;
};

class Obiect : public IImprimabil
{
public:
        void imprima() {…}
};
```

```
//…
void tipareste(const IImprimabil &ii)
{ … }
//…

Obiect o;
tipareste(o);
```

Moștenirea multiplă

- Derivarea din mai multe clase
- Mai multe clase de bază
- Clasele de bază moștenesc membrii claselor de bază din lista de derivare
- Lista de derivare stabilește ordinea apelurilor
  - Constructorilor
  - Destructorilor (ordine inversă)

Moștenirea multiplă

```
class Derivata: [tip_derivare] Baza1, [tip_derivare] Baza2, etc.
{

};
```

Moștenirea multiplă

```
class Imprimanta
{
protected:
        int rezI_x, rezI_y;
public:
        Imprimanta(xi, yi) :
rezI_x(xi), rezI_y(yi) {}
        void tipareste();
};
```

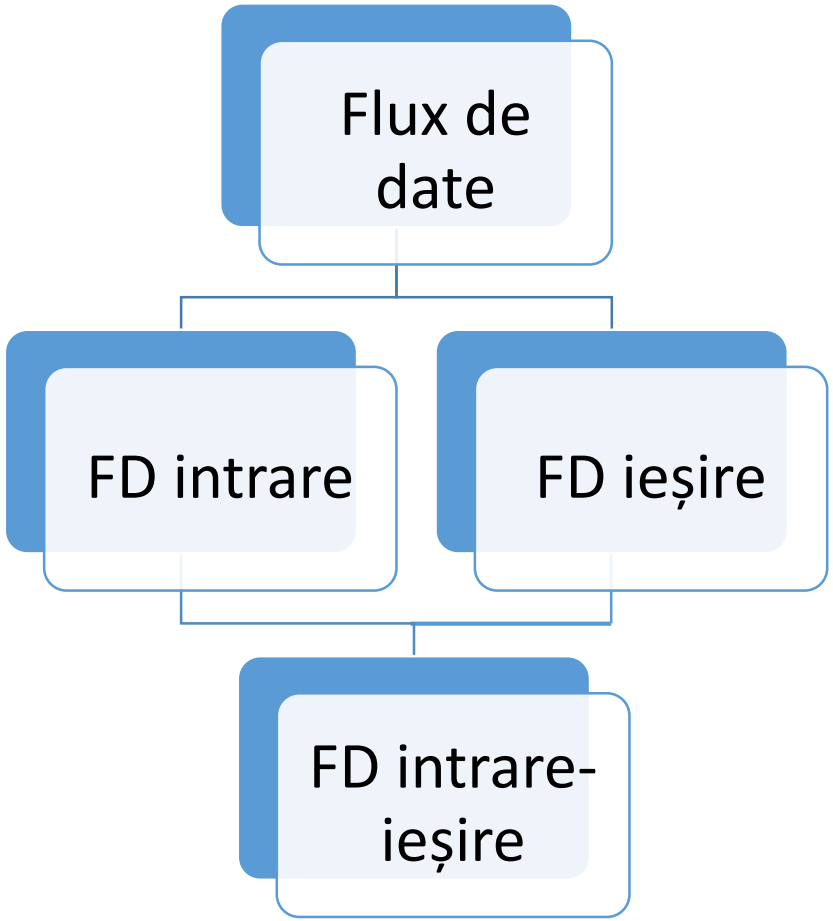
```
class Scanner
{
protected:
        int rezS_x, rezS_y;
public:
        Scanner(xs, ys) : rezS_x(xs),
rezS_y(ys) {}
        void scaneaza();
};
```

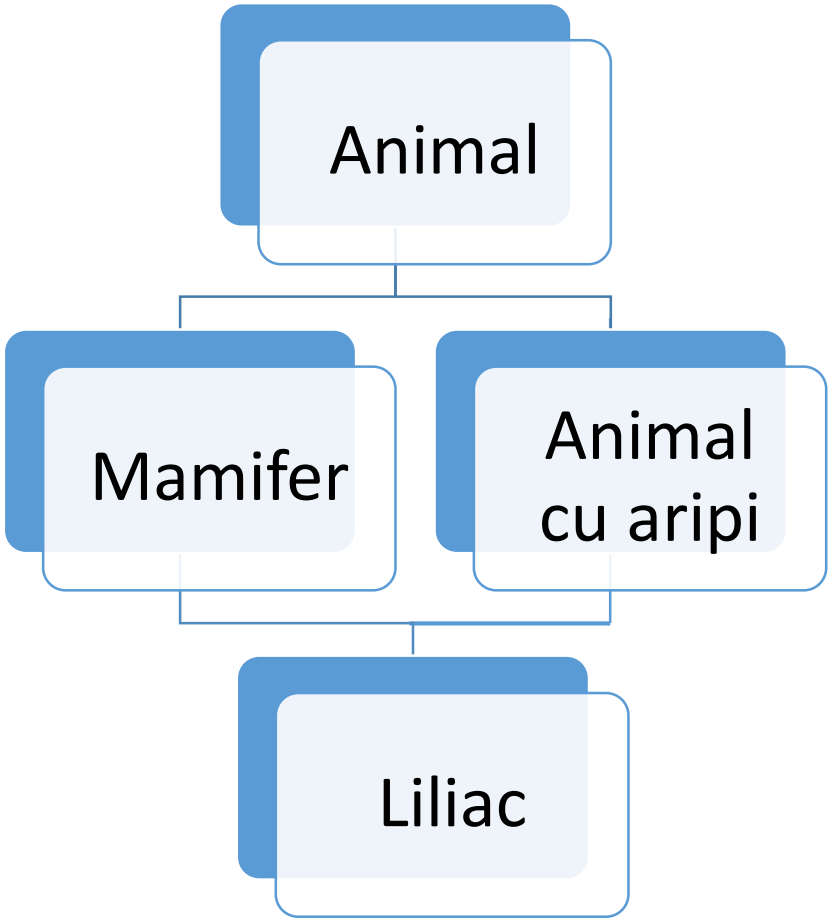
```
class Multifunctional: public Imprimanta, public Scanner
{
public:
        Multifunctional(int xi, int yi, int xs, int ys):
                Scanner(xs, ys), Imprimanta(xi, yi) {}
};
```

Moștenirea multiplă: Ambiguități

1. Problema diamantului
   - Clasele de bază sînt derivate dintr-o clasă comună
   - Clasa derivată cu moștenire multiplă va avea membri duplicat
2. Clasele de bază au membri cu același nume

Moștenirea multiplă: Ambiguități (1)





Moștenirea multiplă: Ambiguități (1)

```
class Animal
{
protected:
        int greutateMed;
public:
        int obtineGreutateaMedie() {…};
};

class Mamifer : public Animal { };
class AnimalCuAripi: public Animal { };
```

```
class Liliac : public Mamifer,
AnimalCuAripi
{ };

void main()
{
   Liliac liliac;
   cout<<liliac.obtineGreutateaMedie();
}
```

Moștenirea multiplă: Ambiguități (1)

```
void main()
{
        Animal *animal;
        animal = &liliac;
}
```

Moștenirea multiplă: Ambiguități (1) – Soluții

- Animal *animal;
- Mamifer *mamifer;
- mamifer = &liliac;
- animal = mamifer;

Moștenirea multiplă: Ambiguități (1) – Soluții

- Referirea explicită a membrului care va fi apelat
  - cout<<liliac.Mamifer::obtineGreutateaMedie()<<endl;
  - cout<<liliac.AnimalCuAripi::obtineGreutateaMedie()<<endl;
- Moștenire virtuală
  - Se moștenește doar o copie a membrului duplicat
  - Pentru toate clasele derivate din clasa comună

Moștenire virtuală

```
class Animal { };
class Mamifer : public virtual Animal { };
class AnimalCuAripi: public virtual Animal { };
class Liliac : public Mamifer, AnimalCuAripi { };
```

Moștenirea multiplă: Ambiguități (2)

```
class Imprimanta
{
public:
        void afiseaza() { }
};

class Scanner
{
public:
        void afiseaza() { }
};
```

```
class MF : public Imprimanta, public
Scanner
{ };

void main()
{
        MF mf;
        mf.afiseaza();
}
```

Moștenirea multiplă: Ambiguități (2) – Soluții

- mf.Scanner::afiseaza();
- mf.Imprimanta::afiseaza();