Universitatea Tehnică „Gheorghe Asachi" din Iași
Facultatea de Automatică și Calculatoare

PROGRAMARE II

Curs 8

Funcții prietene. Clase prietene. Moștenire.

Funcții prietene

- În general, membrii privați sau protejați ai unei clase nu pot fi accesați decât din cadrul clasei
- Funcțiile prietene sunt funcții externe clasei care pot accesa membrii privați
- Sunt declarate utilizând cuvântul cheie `friend`
- Pentru ca o funcție externă clasei să poată accesa membrii privați și protejați se declară prototipul funcției în interiorul clasei, precedat de cuvântul cheie `friend`

Funcții prietene (2)

```
class Data
{
	unsigned char zi, luna;
	int an;
public:
	Data():zi(0),luna(0),an(0){};
	Data(unsigned int z, unsigned int l, int a):
		zi(z),luna(l),an(a){};
};
void afisare (Data d)
{
	cout << (int)d.zi << "-" << (int)d.luna
		<< "-" << d.an << endl;
}
int main ()
{
	Data d1 (14,11,2012);
	afisare (d1);
	return 0;
}
```

```
error C2248: 'Data::zi' : cannot
access private member declared
in class 'Data'
```

Funcții prietene (3)

```
class Data
{
	unsigned char zi, luna;
	int an;
public:
	Data():zi(0),luna(0),an(0){};
	Data(unsigned int z, unsigned int l, int a):
		zi(z),luna(l),an(a){};

	friend void afisare (Data);

};
```

- Funcția afisare nu este membră a clasei Data!

Clase prietene

- Similar definirii unei funcții prietene, se pot defini și clase prietene
- Se garantează clasei prietene accesul la membrii privați și protejați
- Relațiile de prietenie dintre clase
  - nu sunt implicit reciproce
  - nu sunt tranzitive

Exemplu clase prietene

```
class A
{
	int m_ian;
public:
	A(){m_ian=0;};
	void setAn(int anNou)
	{
		m_ian = anNou;
	}
	friend class B;

};
```

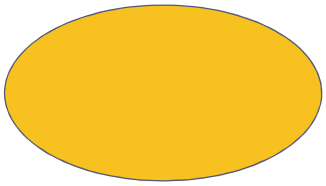
```
class B
{
public:
	B(){};
	void afisare (A objA)
	{
		cout << objA.m_ian;
	}
};
```

```
int main ()
{
	A objA;
	objA.setAn(2001);
	B objB;
	objB.afisare(objA);
	return 0;
}
```

2001

Moştenirea

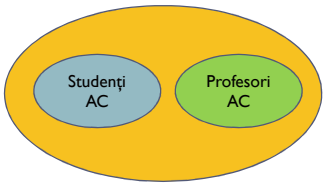
- O clasă defineşte un set de obiecte sau un tip de obiecte



Persoane AC

Tipuri care fac parte dintr-un alt tip

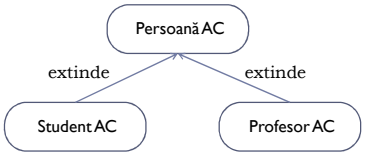
- În cadrul aceluiaşi tip, unele obiecte sunt diferite faţă de altele în anumite privinţe



Persoane AC

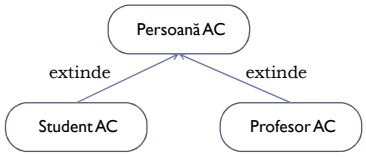
- Profesori AC şi Studenţi AC sunt subtipuri ale Persoane AC

Ierarhii de tipuri



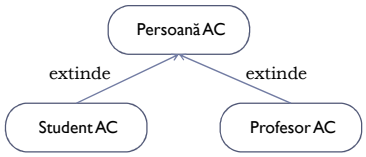
- Ce caracteristici / acţiuni au în comun Persoanele de la AC?
  - Nume, adresă, CNP, etc.
  - Afişare profil, schimbare adresa, etc.

Ierarhii de tipuri



- Ce caracteristici / acțiuni sunt specifice studenților?
  - An de studiu, note la examene, etc.
  - Înscriere într-un an de studiu, amânare, prezentare proiect, etc.

Ierarhii de tipuri



- Ce caracteristici / acțiuni sunt specifice profesorilor?
  - Curs predat, funcție, grad didactic, etc.
  - Promovare, predare curs nou, etc.

Moștenirea

- Un subtip
  - Moștenește caracteristicile și acțiunile tipului din care face parte
  - Are caracteristici și acțiuni specifice
- Fiecare Student AC are:
  - Caracteristici
    - Nume, adresă, CNP, etc.
    - An de studiu, note la examene, etc.
  - Acțiuni
    - Afișare profil, schimbare adresa, etc.
    - Înscriere într-un an de studiu, amânare, prezentare proiect, etc.

Moştenirea

- Fiecare Profesor AC are:
  - Caracteristici
    - Nume, adresă, CNP, etc.
    - Curs predat, funcţie, grad didactic, etc.
  - Acţiuni
    - Afişare profil, schimbare adresa, etc.
    - Promovare, predare curs nou, etc.

Moştenirea C++

- Pentru a deriva o clasă dintr-alta, se foloseşte operatorul ":" la declararea clasei derivate după următorul format:

```
class ClasaDerivata: public ClasadeBaza
{…};
```

- Specificatorul de acces (`public`) poate fi înlocuit de unul dintre ceilalţi doi specificatori `private` sau `protected`

Moştenirea – specificatori de acces

Accesul în clasa de bază	Specificatorul de acces din lista claselor de bază	Accesul în clasa derivată a membrului moştenit
Private	Private	inaccesibil
Protected		private
Public		private
Private	Protected	inaccesibil
Protected		protected
Public		protected
Private	Public	inaccesibil
Protected		protected
Public		public

Clasa de bază PersoanaAC

```
class PersoanaAC
{
protected:
	string m_sCnp;
	string m_sNume;
	string m_sAdresa;
public:
	PersoanaAC(string cnp, string nume, string adresa);
	void afisareProfil();
	void schimbareAdresa(string adresaNoua);
};
```

Clasa StudentAC

```
class StudentAC : public PersoanaAC
{
	int m_ianStudiu;
	int m_inotaP2;
public:
	StudentAC(string cnp, string nume, string adresa, int
anStudiu, int notaP2);
	void afisareProfil();
	void inscriereAnStudiu(int anStudiuNou);
};
```

Constructori

```
PersoanaAC::PersoanaAC(string cnp, string nume, string adresa)
{
	m_sCnp = cnp;
	m_sNume = nume;
	m_sAdresa = adresa;
}


StudentAC::StudentAC(string cnp, string nume, string adresa,
int anStudiu, int notaP2) :
	PersoanaAC(cnp, nume, adresa), m_ianStudiu(anStudiu),
	m_inotaP2(notaP2)
{
}
```

Rescrierea metodelor din clasa de bază

```
void PersoanaAC::afisareProfil()
{
	cout << "Nume: " << m_sNume << " CNP: " << m_sCnp
		<< " Adresa: " << m_sAdresa << endl;
}

void StudentAC::afisareProfil()
{
	cout << "Nume: " << m_sNume << " CNP: " << m_sCnp
		<< " Adresa: " << m_sAdresa << endl;
	cout << "An studiu: " << m_ianStudiu << " Nota P2: " <<
m_inotaP2 <<endl;
}
```

Rescrierea metodelor din clasa de bază

```
void PersoanaAC::afisareProfil()
{
	cout << "Nume: " << m_sNume << " CNP: " << m_sCnp
		<< " Adresa: " << m_sAdresa << endl;
}

void StudentAC::afisareProfil()
{
	PersoanaAC::afisareProfil();
	cout << "An studiu: " << m_ianStudiu << " Nota P2: "
<< m_inotaP2 <<endl;
}
```

```
void PersoanaAC::schimbareAdresa(string adresaNoua)
{
	m_sAdresa = adresaNoua;
}


void StudentAC::inscriereAnStudiu(int anStudiuNou)
{
	m_ianStudiu = anStudiuNou;
}
```

Funcția main

```
int main ()
{

	PersoanaAC p1("1234567890123","Ana","Iasi");
	p1.afisareProfil();

	StudentAC s1("1234567890122","Ion","Vaslui",2,10);
	s1.schimbareAdresa("Bucuresti");
	s1.inscriereAnStudiu(3);
	s1.afisareProfil();

	return 0;
}
```

```
Nume: Ana CNP: 1234567890123 Adresa: Iasi
Nume: Ion CNP: 1234567890122 Adresa: Bucuresti
An studiu: 3 Nota P2: 10
```

Părți din acest curs au fost traduse și adaptate după
MIT OpenCourseWare
http://ocw.mit.edu

6.088 Introduction to C Memory Management and C++ Object-Oriented Programming January IAP 2010