Funcțiile de acces (getteri și setteri)

- Funcții special definite pentru a asigura accesul controlat asupra atributelor private ale clasei
- Asigură acces de tip citire (getter) sau scriere (setter)
- Sunt recomandate în detrimentul atributelor publice deoarece pot asigura accesul read-only asupra lor (prin crearea doar a unui getter pentru acel atribut) sau pot valida valorile ce se dorește a fi salvate în atribute
- Sunt definite în zona publică a clasei

Exemplu getter

```
class TelefonMobil
{
public:
	string getModel()
	{
		return model;
	}
private:
	string model;
protected:
	string producator;
};
```

Utilizare getter

```
int main()
{
	TelefonMobil telefonPersonal("Samsung", "S20");
	string model = telefonPersonal.getModel();
	//va afisa S20
	cout << model << endl;
}
```

Exemplu setter

```
class TelefonMobil
{
public:
	void setModel(string modelNou)
	{
		model = modelNou;
	}
private:
	string model;
protected:
	string producator;
};
```

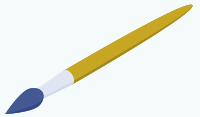
Utilizare setter

```
int main()
{
	TelefonMobil telefonPersonal("Samsung", "S20");
	telefonPersonal.setModel("S20 Ultra");
	//va afisa S20 Ultra
	cout << telefonPersonal.getModel() << endl;
}
```

Pointerul this

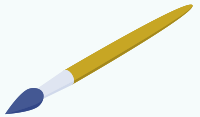
- Utilizat, în special în metode, pentru a accesa obiectul curent (cel prelucrat de acea metodă)
- El salvează adresa acestui obiect
- Este necesară utilizarea acestuia atunci când metoda are un parametru cu aceeași denumire ca un atribut al clasei deoarece variabilele locale (în acest caz parametrul) au prioritate în fața variabilelor globale (în acest caz atributul) cu aceeași denumire

Exemplu



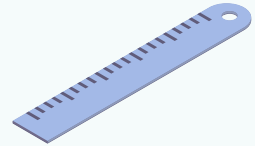
```
class TelefonMobil
{
public:
//...
      TelefonMobil(string producator, string model)
      {
            this->producator = producator;
            this->model = model;
            nivelBaterie = 0;
      }
//...
};
```

Precizare



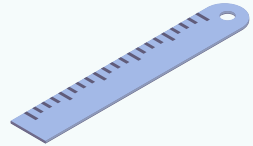
```
int main()
{
	TelefonMobil telefonPersonal("Samsung", "S20");
	//this pentru metoda setModel este adresa obiectului telefonPersonal
	telefonPersonal.setModel("S20 Ultra");
	//de fapt, pentru a putea prelucra obiectul curent, metodele primesc adresa lui
	//ca si prim parametru, astfel incat sa modifice atributele acelui obiect si numai lui
}
```

Metodele inline



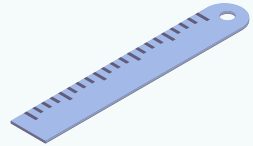
- Metodele unei clase pot fi scrise fie în interiorul clasei în totalitate, fie poate fi scrisă doar signatura lor în clasă, iar corpul în exteriorul clasei prin utilizarea operatorului de rezoluție ::
- De obicei (depinde de compilator) metodele scrise în interiorul clasei sunt considerate metode inline
- Metodele scrise în afara clasei pot fi transformate în metode inline prin folosirea cuvântului cheie **inline** (acest lucru va duce la mutarea lor automată în clasă în faza de preprocesare, asemănător cu ce se întâmplă cu #define)

Exemplu



```
class TelefonMobil
{
float nivelBaterie;
	//metoda inline
	void incarca(int nivelIncarcare)
	{
		nivelBaterie += nivelIncarcare;
	}
	void descarca(int nivelDescarcare);
};

void TelefonMobil::descarca(int nivelDescarcare)
{
	nivelBaterie -= nivelDescarcare;
}
```

Transformarea în metodă inline



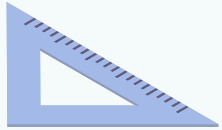
```
class TelefonMobil
{
float nivelBaterie;
	//metoda inline
	void incarca(float nivelIncarcare)
	{
		nivelBaterie += nivelIncarcare;
	}
	void descarca(float nivelDescarcare);
};

inline void TelefonMobil::descarca(float nivelDescarcare)
{
	nivelBaterie -= nivelDescarcare;
}
```

Curs 05

TIPURI DE MEMBRI

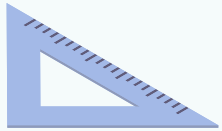
Membri constanți, statici sau pointeri
Compunerea claselor și fișierele header

Membri constanți



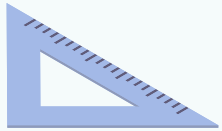
- Atributele unei clase sunt, mai mult sau mai puțin, echivalentul unor variabile
- Fiecare dintre atribute poate lua mai multe valori de-a lungul existenței unui obiect
- Cu toate acestea există situații când vrem ca valoarea unui atribut să nu se mai schimbe odată ce l-am inițializat
- Pentru a realiza acest lucru se pot defini atribute constante

Exemplu atribut constant



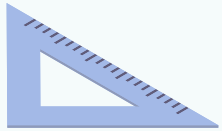
```
class TelefonMobil
{
public:
	const int durataMaximaBaterie = 12;
//...
};

int main()
{
	TelefonMobil telefonPersonal;
	//eroare de compilare
	//telefonPersonal.durataMaximaBaterie = 20;
}
```

Exemplu membru constant



```
class TelefonMobil
{
public:
	const int durataMaximaBaterie;
	TelefonMobil() : durataMaximaBaterie(9)
	{
		//...
	}
	TelefonMobil(string producator, int durataMaximaBaterie) : durataMaximaBaterie(durataMaximaBaterie)
	{
		this->producator = producator;
	}

private:
	string producator;
//...
};
```

Exemplu membru constant



```
int main()
{
	//durataMaximaBaterie va avea valorea 9
	TelefonMobil telefonPersonal;
	//eroare de compilare
	//telefonPersonal.durataMaximaBaterie = 20;
	TelefonMobil telefonDeServiciu("Nokia", 48);
	//eroare de compilare, duraraBaterie va ramane 48 cat timp exista obiectul
	//telefonDeServiciu.durataMaximaBaterie = 20;
}
```

Membri statici

- Atributele sau metodele despre care am discutat până acum sunt specifice fiecărui obiect în parte
- Cu toate acestea, sunt situații când dorim ca anumite atribute sau metode să nu fie specifice obiectului, ci clasei
- Exemple: un atribut care este comun tuturor obiectelor, o metodă ce nu prelucrează doar un obiect

Atributele statice

- Se definesc folosind cuvântul cheie **static** și sunt atribute ale clasei, nu ale obiectului
- Valoarea atributului este comună pentru toate obiectele (odată schimbată valoarea pentru un obiect se schimbă pentru toate celelalte)
- A nu se confunda cu variabilele statice din C (variabile globale ce se pot defini și în interiorul funcției) sau cu vectorii statici (vectori ce nu sunt alocați dinamic, a căror dimensiune este cunoscută la compilare)

Exemplu atribut static

```
class TelefonMobil
{
public:
	static int anulFabricatiei;

private:
	string producator;
//...
};

int TelefonMobil::anulFabricatiei = 2020;
```

Exemplu atribut static

```
int main()
{
	TelefonMobil telefonPersonal;
	//vor afisa 2020
	cout << telefonPersonal.anulFabricatiei;
	cout << TelefonMobil::anulFabricatiei;
	telefonPersonal.anulFabricatiei = 2021;
	TelefonMobil telefonDeServiciu("Nokia", 48);
	//va afisa 2021
	cout << telefonDeServiciu.anulFabricatiei;
}
```

Metodele statice

- Se definesc folosind cuvântul cheie **static**
- Sunt metode ce fac referire la clasă și nu la obiect (asemănătoare unor metode globale)
- Nu primesc pointerul **this** și nu pot prelucra atributele non-statice ale clasei
- Exemple: funcții ce prelucrează atributele statice (inclusiv getteri și setteri), metode ce prelucrează vectori de obiecte, etc.

Exemplu metodă statică

```
class TelefonMobil
{
public:
	static float NivelMediuBaterie(TelefonMobil* telefoane, int nrTelefoane)
	{
		//...
	}
private:
	string producator;
//...
};
```

Exemplu metodă statică

```
int main()
{
	TelefonMobil vectorTelefoane[] = { t1, t2, t3 };
	float medie = TelefonMobil::NivelMediuBaterie(vectorTelefoane, 3);
	cout << medie;
}
```