

8. lekce

Dědičnost objektů

Dědičnost objektů

- Jeden objekt zdědí některé (*nebo všechny*) atributy (*strukturu dat*) a metody (*konstruktor, destruktor, kontroly, přetěžování operátorů, ...*) od objektu druhého.
- Výhodou je to, že u potomků nemusíme definovat funkce, které zdědí od rodičů.
- Pokud funkci definujeme v potomkovi a zároveň je tato funkce definována v rodiči, pak je vždy použita funkce z potomka.
- Atributy a metody, u kterých chceme, aby je potomek zdědil, musí být v rodiči definovány v sekci **public** nebo **protected**.
Ty, které jsou v sekci **private**, ty zdědit nelze (potomci „vidí“ do **protected**, ale nevidí do **private**).
- Instance (*jedna konkrétní hodnota objektu – např. zlomek 1/4, student Miroslav Jílek*) potomka může být přiřazena rodiči, ale instance rodiče nemůže být přiřazena potomkovi. Důvodem je to, potomek je „větší“ - má zpravidla svoje další atributy, které nelze vložit do rodiče.

Způsob vytvoření objektu, který dědí:

Objekt **Rodic** a z něho vytvoříme objekt **Potomek**:

```
class Potomek: Rodic
{
    atributy a metody objektu
}
```

*Takto se zdědí všechno, co bylo v public a protected objektu Rodic.
To, co nechceme, aby Potomek od Rodice zdědil, definujeme v Rodici v private.*

Potomek

– *Zachování viditelnosti členů rodiče*

```
class Potomek: public Rodic    // v běžných případech používáte public
{
    atributy a metody objektu
}
```

Všechno, co bylo v Rodic public, bude v Potomek public. Všechno co bylo v Rodic protected, bude v Potomek také protected.

Část definovaná v private se nedědí a v Potomek si ji definuje nově (pokud potřebujeme!), případně si definujeme atributy pro objekt Potomek

– *Změna viditelnosti všech členů rodiče na private v potomkovi*

```
class Potomek: private Rodic
{
    atributy a metody objektu
}
```

Všechno, co bylo v Rodic v public a protected, bude v Potomek v private.

Příklad:

```
class clovek
{
protected:
    string m_jmeno;
    string m_prijmeni;
public:
    clovek(string jmeno, string prijmeni): m_jmeno(jmeno),
                                           m_prijmeni(prijmeni)
    {
        if (jmeno=="nikdo") throw "Chyba"; //kontrola se přenesse do student
    }
};

class student: public clovek
{
private:
    string m_zeme;
public:
    student(string jmeno, string prijmeni, string zeme): clovek(jmeno, prijmeni),
                                                         m_zeme(zeme)
    {}
};
```

Když konstruujeme potomka, musím nejprve konstruovat rodiče – proto:

student(string jmeno, string prijmeni, string zeme): clovek(jmeno, prijmeni), m_zeme(zeme)