Objektově orientované programování

Návrh programu I

2023/24

Osnova přednášky

- Co víme?
- Objektový návrh programu.
- Příklad.

Co víme?

Třída

- Třída je popisem objektů se společnými vlastnostmi.

class <jméno>
    private:
        <soukromé členské položky>
    public:
        <veřejné členské položky>

- Členskými položkami mohou být proměnné (data) a metody (funkce).

Objekt

- Objekt je instancí - paměťovou reprezentací – třídy.
- Objekt je reprezentací nějaké entity, která má stav reprezentovaný daty a chování reprezentované metodami.

Konstruktor

- Konstruktor inicializuje objekty.
- Nemá návratovou hodnotu.
- Pokud není deklarován, automaticky se vytvoří (implicitní) konstruktor.
- Volá se automaticky při statické deklaraci nebo použitím **new**.
- *Konstruktorů může být více, musí se lišit počtem nebo typem parametrů.*

Destruktor

- Slouží pro dealokaci paměti vytvořené dynamicky.
- Nemá návratovou hodnotu.
- Pokud není deklarován, automaticky se vytvoří (implicitní) destruktor.
- Volá se automaticky při statické deklaraci nebo použitím **delete**.

Objektový návrh

Zadání

- Mějme malou banku s omezeným počtem klientů a účtů. V bance mohou klienti a účty přibývat.
- Každý účet má jednoho vlastníka a může mít jednoho partnera, oba jsou klienti banky a mají jméno a kód. Na účty lze vkládat a vybírat z nich, lze zjistit stav na účtu. Pokud není na účtu dostatek peněz, nelze vybrat.
- Vklady na účtech jsou úročeny, a to buď základní nebo speciální úrokovou sazbou. Jednou za čas banka všem účtům připíše úrok odpovídající úrokové sazbě.
- Účet resp. klienta je možno v bance vyhledat podle čísla resp. kódu.
- Lze zjistit, ale nelze měnit:
  - číslo účtu a jeho úrokovou sazbu
  - kód a jméno klienta,
  - vlastníka a partnera účtu.

Třídy

- Mějme malou **banku** s omezeným počtem **klientů** a **účtů**. V bance mohou klienti a účty přibývat.

- Každý účet má jednoho vlastníka a může mít jednoho partnera, oba jsou klienti banky a mají jméno a kód. Na účty lze vkládat a vybírat z nich, lze zjistit stav na účtu. Pokud není na účtu dostatek peněz, nelze vybrat.

- Vklady na účtech jsou úročeny, a to buď základní nebo speciální úrokovou sazbou. Jednou za čas banka všem účtům připíše úrok odpovídající úrokové sazbě.

- Účet resp. klienta je možno v bance vyhledat podle čísla resp. kódu.

Chování

- Mějme malou banku s omezeným počtem klientů a účtů. V bance mohou klienti a účty **přibývat**.
- Každý účet má jednoho vlastníka a může mít jednoho partnera, oba jsou klienti banky a mají jméno a kód. Na účty lze **vkládat** a **vybírat** z nich, lze **zjistit stav** na účtu. Pokud není na účtu dostatek peněz, nelze vybrat.
- Vklady na účtech jsou úročeny, a to buď základní nebo speciální úrokovou sazbou. Jednou za čas banka všem účtům **připíše úrok** odpovídající úrokové sazbě.
- Účet resp. klienta je možno v bance **vyhledat** podle čísla resp. kódu.

Stav

- Mějme malou banku s omezeným počtem klientů a účtů. V bance mohou **klienti** a **účty** přibývat.
- Každý účet má **jednoho vlastníka** a může mít **jednoho partnera**, oba jsou klienti banky a mají **jméno** a **kód**. Na účty lze vkládat a vybírat z nich, lze zjistit **stav na účtu**. Pokud není na účtu dostatek peněz, nelze vybrat.
- Vklady na účtech jsou úročeny, a to buď základní nebo speciální **úrokovou sazbou**. Jednou za čas banka všem účtům připíše úrok odpovídající úrokové sazbě.
- Účet resp. klienta je možno v bance vyhledat podle čísla resp. kódu.

Třída *Client*

- Kód a jméno.
- Lze se na ně zeptat.
- Nelze je měnit.

Třída *Account*

- Číslo a částka, vlastník a partner, úroková sazba.
- Lze se na ně zeptat.
- Číslo, úrokovou sazbu, vlastníka a partnera nelze měnit.
- Vložit, vybrat, zjistit stav, připsat úrok.
- Nemusí jít vybrat.

Třída *Bank*

- Omezený seznam klientů a účtů.
- Lze přidat nového klienta a účet.
- Lze hromadně připsat úrok dle úrokové sazby.
- Lze vyhledat klienta podle kódu a účet podle čísla.

Příklad

Deklarace

```
class Client
{
private:
    int code;
    string name;

public:
    Client(int c, string n);

    int GetCode();
    string GetName();
};
```

```
class Account
{
private:
    int number;
    double balance;
    double interestRate;

    Client *owner;
    Client *partner;

public:
    Account(int n, Client *c);
    Account(int n, Client *c, double ir);
    Account(int n, Client *c, Client *p);
    Account(int n, Client *c, Client *p, double ir);

    int GetNumber();
    double GetBalance();
    double GetInterestRate();
    Client *GetOwner();
    Client *GetPartner();
    bool CanWithdraw(double a);

    void Deposit(double a);
    bool Withdraw(double a);
    void AddInterest();
};
```

```
class Bank
{
private:
    Client** clients;
    int clientsCount;

    Account** accounts;
    int accountsCount;

public:
    Bank(int c, int a);
    ~Bank();

    Client* GetClient(int c);
    Account* GetAccount(int n);

    Client* CreateClient(int c, string n);
    Account* CreateAccount(int n, Client *c);
    Account* CreateAccount(int n, Client *c, double ir);
    Account* CreateAccount(int n, Client *c, Client *p);
    Account* CreateAccount(int n, Client *c, Client *p, double ir);

    void AddInterest();
};
```

Tři typy metod

- **Konstruktory a destruktor.** Konstruktory inicializují stav objektu po jeho vzniku, destruktor uvolňuje dynamicky přidělenou paměť před zánikem objektu. Pokud nejsou uvedeny, vytvoří se tyto metody automaticky (bez algoritmu).

- **Metody poskytující informaci o stavu objektu.** Metody buď podávají informaci přímo o hodnotě datové položky objektu nebo poskytují informaci na základě algoritmu.

- **Metody měnící stav objektu.** Metody buď přímo změní hodnotu datové položky objektu nebo provedou změnu na základě algoritmu.

- *Metody poskytující informaci o stavu objektu by neměly jeho stav měnit!!!*

Objektové kompozice

- Objekt se může stát součástí jiného objektu a stává se tak jeho datovou položkou.
- Vznikají tak komponované objekty s přesně definovanými kompetencemi. Ty ovšem mohou realizovat prostřednictvím interakce objektů, ze kterých jsou komponovány.
- Například *účet* má *klienty* v rolích vlastníka a partnera resp. *banka* má *klienty* a *účty*.
- Jeden a tentýž objekt může být součástí více kompozic. Například jeden *klient* může být součástí jak *účtu*, tak *banky*.

Úkoly na cvičení

- Implementujte příklad z přednášky a navrhněte kód, který bude používat všechny třídy. Vytvořte desítky klientů a účtů v bance a nasimulujte některé běžné úkony prováděné v bance.
- Navrhněte a implementujte podobnou úlohu, jako například lékařskou ordinaci, malou školu apod.

Kontrolní otázky

- Vysvětlete, jak vznikají objekty třídy, pojem konstruktor a principy práce s ním v C++.
- Vysvětlete, jak zanikají objekty třídy, pojem destruktor a principy práce s ním v C++.
- Vysvětlete rozdíl mezi statickou a dynamickou deklarací objektů v C++.
- Jak se dá postupovat, pokud chceme v zadání programu nalézt třídy, jejich metody a datové členy?
- Kdy a proč potřebujeme použit více konstruktorů jedné třídy?
- Kdy potřebujeme deklarovat a definovat destruktor?
- Co jsou výchozí konstruktory a destruktory a k čemu je potřebujeme?
- Jaké typy metod obvykle musíme deklarovat a definovat?
- Co jsou objektové kompozice a k čemu jsou dobré?