e-Banking

Termín odevzdání:	**12.04.2020 23:59:59**
Pozdní odevzdání s penalizací:	**30.06.2020 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)
Hodnocení:	**5.5055**
Max. hodnocení:	**7.1500** (bez bonusů)
Odevzdaná řešení:	1 / 20 Volné pokusy + 20 Penalizované pokusy (-2 % penalizace za každé odevzdání)
Nápovědy:	0 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je realizovat třídu `CBank`, která bude implementovat internetové bankovnictví.

Uvažovaná třída je specifická tím, že chceme ukládat všechny změny (všechny transakce). Dále, pro účely archivace chceme ukládat nezávislé kopie instancí.

Úloha je směřována k procvičení pochopení mělké a hluboké kopie objektu. Má části povinné, dobrovolné a bonusové. Pro splnění povinné části úlohy postačuje vytvořit funkční třídu, která bude splňovat požadované rozhraní. Pro zvládnutí nepovinných a bonusových částí je potřeba rozmyslet ukládání dat tak, aby se při kopírování zbytečně neplýtvalo pamětí.

Úloha má procvičit práci s kopírováním objektů. Z tohoto důvodu jsou v úloze potlačené části standardní C++ knihovny, zejména STL a datový typ C++ string.

Požadovaná třída `CBank` má následující rozhraní:

`implicitní konstruktor`
: vytvoří prázdnou instanci el. bankovnictví,

`kopírující konstruktor`
: vytvoří hlubokou kopii instance. Implementujte jej, pokud kompilátorem automaticky vytvořený kopírující konstruktor nevyhovuje (zkuste si odhadnout - bude potřeba?),

`destruktor`
: uvolní prostředky alokované instancí.

`operátor =`
: zkopíruje obsah jedné instance do druhé (hluboká kopie). Implementujte jej, pokud kompilátorem automaticky vytvořený operátor = nevyhoví,

`NewAccount(accID,initBalance)`
: metoda přidá nové konto:

  - `accID` udává jednoznačný identifikátor konta. Jedná se o řetězec, délka a obsah může být libovolný, je požadována pouze unikátnost,
  - `initBalance` udává počáteční vklad (celé číslo). Implementovaná banka je velmi vstřícná, a připouští i záporný počáteční vklad.

  Metoda `NewAccount` vrací hodnotu `true` pro signalizaci úspěchu, `false` pro neúspěch (konto se stejným `id` již bylo zadané).

`Transaction(debitAcc,creditAcct,amount,sign)`
: metoda realizuje bezhotovostní převod z jednoho konta na druhé. Parametry jsou:

  - `debitAcc` udává identifikátor konta plátce (z něj se částka odečte),
  - `creditAcc` udává identifikátor konta příjemce (na něj se částka přičte),
  - `amount` udává převáděnou částku (může být i nulová),
  - `sign` udává elektronický podpis autorizace převodu. Jedná se o řetězec, který je potřeba uložit pro účely dokladovaní (aplikace s ním jinak nemusí nijak pracovat).

  Metoda `Transaction` vrací hodnotu `true` pro signalizaci úspěchu, `false` pro neúspěch (zadané konto neexistuje). Metoda nekontroluje zůstatek na účtu, předpokládáme, že lze konto bez problémů přečerpat. Převod ale selže, pokud se konto plátce shoduje s kontem příjemce.

`TrimAccount(accID)`
: metoda odmaže archivované transakce z daného konta. Tedy aktuální zůstatek na kontě se stane počátečním stavem a archivované transakce zmizí. Volání lze chápat tak, že klient potvrdil předchozí transakce, tedy nemá cenu je dále archivovat. Metoda `TrimAccount` vrací hodnotu `true` pro signalizaci úspěchu, `false` pro neúspěch (zadané konto neexistuje).

`cout << Account(accID)`
: výstup metody `Account` zaslaný do streamu zobrazí počáteční stav konta, seznam archivovaných transakcí a konečný stav konta. Formát je zřejmý z ukázky. Pokud účet zadaného čísla neexistuje, je vyhozena (libovolná) výjimka.

`Account(accID) . Balance()`
: výstup metody `Account` musí reagovat na volání `Balance`, toto volání vrátí konečný stav zadaného konta. Pokud účet zadaného čísla neexistuje, je vyhozena (libovolná) výjimka.

Odevzdávejte soubor, který obsahuje implementovanou třídu `CBank` a další Vaše podpůrné třídy. Třída musí splňovat veřejné rozhraní podle ukázky - pokud Vámi odevzdané řešení nebude obsahovat popsané rozhraní, dojde k chybě při kompilaci. Do třídy si ale můžete doplnit další metody (veřejné nebo i privátní) a členské proměnné. Odevzdávaný soubor musí obsahovat jak deklaraci třídy (popis rozhraní) tak i definice metod, konstruktoru a destruktoru. Je jedno, zda jsou metody implementované inline nebo odděleně. Odevzdávaný soubor nesmí obsahovat vkládání hlavičkových souborů a funkci `main`. Funkce `main` a vkládání hlavičkových souborů může zůstat, ale pouze obalené direktivami podmíněného překladu jako v ukázce (přiložený archiv).

Pokud se rozhodnete řešit i nepovinné a bonusové části úlohy, můžete pro nalezení vhodné reprezentace využít následující pozorování:

- Vzniká mnoho kopií instance, které ale dále nejsou měněné. Tedy vyplatí se uvažovat realizaci, kde jsou data sdílená s originálem dokud nedojde k první modifikaci originálu nebo kopie.
- Změny v instanci jsou "malé", tedy velká část obsahu zůstane stejná jako v originálu. Vyplatí se tedy i uvažovat o sdílení nezměněných částí instance. Pokud byste si např. konta realizovali jako třídy, dá se předpokládat, že mezi originálem a kopií se změní pouze několik kont.

Rozhraní a ukázka použití třídy je dostupná v přiloženém archivu.

Funkční řešení této domácí úlohy může být použito pro code review (řešení musí projít povinnými a nepovinnými testy na 100%, nemusí projít bonusovými testy).

Vzorová data: **Download**

☐ **Referenční řešení**

1	**03.04.2020 23:25:10**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	5.5055	

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test s parametry podle ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.000 s (limit: 10.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.160 s (limit: 10.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test kopirujiciho konstruktoru': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.188 s (limit: 9.840 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test operatoru =': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.218 s (limit: 9.652 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi hodnotami + mem debugger': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.498 s (limit: 10.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Efektivita - kopie beze zmen': Neúspěch
    - Dosaženo: 0.00 %, požadováno: 70.00 %
    - Celková doba běhu: 0.495 s (limit: 5.000 s)
    - Neúspěch v nepovinném testu, hodnocení: 70.00 %
    - Nesprávný výstup **[Zpřístupnit nápovědu (73 B)]**
  - Test 'Efektivita - kopie + malo zmen': Neúspěch
    - Dosaženo: 0.00 %, požadováno: 100.00 %
    - Celková doba běhu: 3.088 s (limit: 5.000 s)
    - Neúspěch v bonusovém testu, hodnocení: Bonus nebude udělen
    - Nesprávný výstup **[Zpřístupnit nápovědu (72 B)]**

    - Celkové hodnocení: 70.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 0.70)
- Celkové procentní hodnocení: 70.00 %
- Bonus za včasné odevzdání: 0.72
- Celkem bodů: 0.70 * (7.15 + 0.72) = 5.51

		Celkem	Průměr	Maximum	Jméno funkce
SW metriky:	Funkce:	**28**	--	--	--
	Řádek kódu:	**352**	**12.57 ± 26.66**	**150**	`main`
	Cyklomatická složitost:	**41**	**1.46 ± 0.82**	**4**	`CBank::Transaction`