Tel-102 Paralelo 2
2º Semestre 2020
Control 2
07 de Octubre de 2020

Instrucciones

- Lea con detenimiento y desarrolle **individualmente** cada una de las actividades a realizar durante la experiencia.
- Cree un archivo con extensión **.cpp** con lo desarrollado. El nombre del archivo debe tener el siguiente formato: **TEL102_C3_Nombre_Apellido.cpp** (Ej. TEL102_C3_Patricio_Olivares.cpp), sin incluir tíldes.
- Enviar el archivo a través de la página de aula del ramo, sección "Control 3" hasta las 11:45:00 del día de **hoy**, Martes 03/11/2020, Hora continental de Chile (UTC-3).
- Cada minuto de atraso tendrá un descuento siguiendo la serie de Fibonacci.
- Trate de utilizar herramientas conocidas o aprendidas en clases. **No copie literalmente de recursos online**.
- Comente adecuadamente el programa, describiendo lo que hace.
- Sea riguroso con las instrucciones de desarrollo.
- ¡Éxito!

Tel-Bank

El nuevo banco **Tel-Bank** ha abierto sucursales en Chile. En este nuevo contexto de pandemia, necesita de un robusto sistema informático para manejar las cuentas de sus clientes. Para esto, ha solicitado sus servicios como programador.
Se le entrega una porción de código **control3.cpp**, (disponible en la página de Aula), el cual será la base para su programa. Este código contiene las clases

- `Cuenta`, la cual cuente información sobre cada cuenta bancaria
- `Banco`, la cual contiene en un arreglo, todas las cuentas asociadas al banco y las transacciones entre ellas.

control3.cpp

```
#include<iostream>
using namespace std;

// Las tildes han sido omitidas

const int cobro = 1000; // Cobro por transaccion
const int n_cuentas = 4; // Numero de cuentas

class Cuenta{
    public:
        Cuenta(int idCuenta, int saldo);
        int getId();
        int getSaldo();
        void depositar(int d); // Deposita cantidad d de dinero
        bool retirar(int d); // Retira cantidad d de dinero
    private:
        int idCuenta;
        int saldo;
};

class Banco{
    public:
        Banco(int n_cuentas, Cuenta *cuentas);
        ~Banco();
        bool transaccion(int id1, int id2,int d); // Transaccion entre cuentas con d dinero
        void estadoBanco(); // Muestra cuentas del banco
    private:
        Cuenta *cuentas;
        int n_cuentas;
        int ganancias;
};
```

1. (34pts) Implemente el constructor y los métodos de la clase `Cuenta`. Considere lo siguiente:
   - Existe un único constructor, con dos párametros `idCuenta` y `saldo`, que sirven para inicializar los argumentos de la clase que poseen los mismos nombres.
   - Los métodos `getId` y `getSaldo`, sirven para **retornar** los valores actuales de los parámetros correspondientes por el nombre de la función.
   - Los métodos `depositar` y `retirar`, sirven para incrementar y reducir el saldo de una beca, respectivamente.
   - Además, el banco (metodo `retirar`) no permite endeudarse, por lo que el saldo de la cuenta **no puede ser negativo**. Si se intenta retirar más dinero del que se tiene, se retorna `false`; y en caso contrario, se retorna `true`.
2. (33pts) Implemente el constructor, el destructor y los métodos de la clase `Banco`. Considere lo siguiente:
   - Existe un único constructor, con dos parámetros `n_cuentas` y `cuentas`, que sirven para inicializar los argumentos de la clase que poseen los mismos nombres de variables. Mientras `n_cuentas` es la cantidad de cuentas del banco, `cuentas` es un arreglo con los datos de dichas cuentas. Considere que al inicio, el Banco no genera ganancias.
   - Existe un destructor que debe evitar fugas de memoria.
   - El método `transaccion` realiza una transferencia de saldo `d` desde una cuenta de origen (`id1`) a una cuenta de destino (`id2`). Si la transacción pudo efectuarse, se retorna `true`, en caso contrario `false`.

- Por cada transacción realizada, el banco gana el monto indicado en la variable `cobro`. Este monto se descuenta de la cuenta de origen `id1`.
- El método `estadoBanco`, debe mostrar por pantalla cada una de las distintas cuentas (`id`) del banco y su respectivo `saldo`.

3. (33pts) Implemente la función `main`. Esta función debe

   - Crear 4 cuentas con los siguientes datos:
     - id = 1124, saldo = 200000
     - id = 5748, saldo = 50000
     - id = 9935, saldo = 30000
     - id = 7548, saldo = 100000
   - Cree un banco con las cuentas anteriormente creadas y sin ganancias.
   - Muestre por pantalla el estado inicial de todas las cuentas y las ganancias iniciales.
   - Realice (en orden) las siguientes transacciones:
     - Transacción 1: id Origen = 5748, id Destino = 1124, monto = 30000
     - Transacción 2: id Origen = 1124, id Destino = 9935, monto = 10000
     - Transacción 3: id Origen = 7548, id Destino = 1124, monto = 15000
     - Transacción 4: id Origen = 5748, id Destino = 9935, monto = 20000

     e indique si cada una de ellas se puede realizar.
   - Muestre por pantalla el estado final de todas las cuentas y sus ganancias.

Salida (consola)

```
[elprofe@tel102 control3]$ ./TEL102_Nombre_Apellido
Cuentas creadas:

Cuenta id:1124
Saldo: 200000
Cuenta id:5748
Saldo: 50000
Cuenta id:9935
Saldo: 30000
Cuenta id:7548
Saldo: 100000
Ganancias: 0

Transaccion 1 realizada correctamente
Transaccion 2 realizada correctamente
Transaccion 3 realizada correctamente
Transaccion 4 no es posible de realizar
Transacciones finalizadas

Cuenta id:1124
Saldo: 234000

Cuenta id:5748
Saldo: 19000

Cuenta id:9935
Saldo: 40000

Cuenta id:7548
Saldo: 84000

Ganancias: 3000
```