ESTRUCTURAS ABSTRACTAS DE DATOS Y ALGORITMOS PARA INGENIERÍA IE0217

AVANCE PROYECTO FINAL

Rubén García Méndez
Aurelio Córdoba Valerio
Óscar Fuentes Córdoba

AVANCES EN CLASES HASTA EL MOMENTO

Clase préstamos

Tiene la propiedad de calcular las cuotas mensuales, su método principal hace este cálculo, es necesario unir a la clase de cuenta para que se hagan los prestamos para el cliente en particular

Clase cuenta

Tiene como atributos el tipo de moneda y saldo y la tasa de cambio como también métodos principales.

Clase cliente

La clase cliente usa los métodos para depositar y retirar, como también describe el método solicitar préstamo y se visualiza los detalles del cliente.

Clase CDP

Se creó la clase con los atributos: interés, monto y plazo, y métodos informativos.

AVANCES GENERALES

Construcción del menú principal

Se completó el diseño del menú principal que asegura el flujo de la información a través del programa, se tiene un buen manejo de errores, pero se puede mejorar.

Creación de los archivos csv

Se definió un formato de trabajo para mantener un orden en los números de identificación de usuarios y préstamos.

Sistema de identificación de clientes

Se está trabajando en el sistema de identificación de clientes utilizando como base de datos un archivo csv.

Información general de préstamos

Se completó la opción para desplegar los préstamos disponibles para los clientes con su respectiva información.

FORMATO PARA IDENTIFICACION DE PRESTAMOS

Dígito número 1

Moneda
0 - Colones
1 - Dólares

Dígito número 2

Tipo de préstamo
0 - Personal
1 - Hipotecario
2 - Prendario

Dígitos del 3-6

Número de identificación de clientes

Dígitos del 7 en adelante

Número del préstamo

Por ejemplo: 1-2-6412-15

Clase Cliente

METODO PARA SOLICITAR PRESTAMOS

```cpp
// Método para solicitar préstamos
void Cliente::solicitarPrestamo(double monto) {
    if (monto > 0) {
        // Aquí se añade lógica para solicitar y aprobar el préstamo
        std::cout << "Préstamo solicitado con éxito, pendiente de aprobación." << std::endl;
    } else {
        std::cerr << "Error: El monto del préstamo debe ser positivo." << std::endl;
    }
}
```

METODO PARA RETIRAR

```cpp
// Métodos para retirar de cuentas
bool Cliente::retirarDeCuentaColones(double cantidad) {
    if (cantidad > 0) {
        if (cuentaColones.retirar(cantidad)) {
            std::cout << "Retiro realizado con éxito de cuenta de colones." << std::endl;
            return true;
        } else {
            std::cout << "Error: Saldo insuficiente para el retiro." << std::endl;
        }
    } else {
        std::cerr << "Error: La cantidad a retirar debe ser positiva." << std::endl;
    }
    return false;
}
```

METODO PARA DEPOSITAR

```cpp
// Métodos para depositar en cuentas ya sea en colones o en dolares
void Cliente::depositarEnCuentaColones(double cantidad) {
    if (cantidad > 0) {
        cuentaColones.depositar(cantidad);
        std::cout << "Depósito realizado con éxito en cuenta de colones." << std::endl;
    } else {
        std::cerr << "Error: La cantidad a depositar no puede ser negativa." << std::endl;
    }
}
```

Clase Cuenta

TRANSACCION ENTRE CUENTAS

```cpp
// Transacción entre cuentas
bool Cuenta::transaccion(double cantidad, Cuenta &cuentaDestino) {
    if (!retirar(cantidad)) return false;
    cuentaDestino.depositar(cantidad);
    std::cout << "Transacción realizada con éxito." << std::endl;
    return true;
}
```

CONVERTIR SALDO

```cpp
// Convertir saldo entre monedas
void Cuenta::convertirSaldo(const std::string& nuevaMoneda) {
    if (tasaDeCambio <= 0) {
        std::cerr << "Error: La tasa de cambio no ha sido establecida." << std::endl;
        return;
    }
    if (tipo == nuevaMoneda) {
        std::cout << "La cuenta ya está en " << nuevaMoneda << "." << std::endl;
        return;
    }
    saldo = (nuevaMoneda == "Dolares" && tipo == "Colones") ? saldo / tasaDeCambio : saldo * tasaDeCambio;
    tipo = nuevaMoneda;
    std::cout << "Conversión realizada. Nuevo saldo: " << saldo << " " << tipo << "." << std::endl;
}
```

CONSULTAR SALDO

```cpp
// Consultar saldo
double Cuenta::consultarSaldo() const {
    return saldo;
}
```

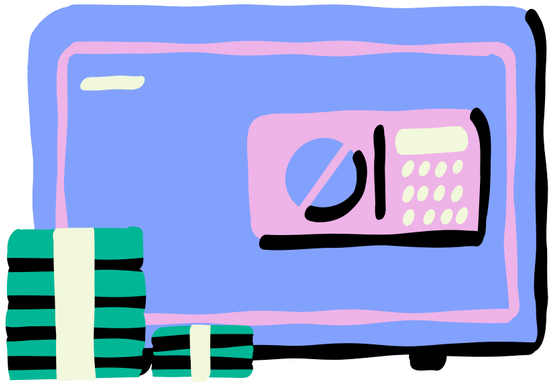
Clase Préstamo

MÉTODO DE CALCULO DE PRESTAMO

```cpp
Prestamo::Prestamo(Tipo tipo, double tasaInteresAnual, int cantidadCuotas, double montoPrestamo)
    : tipoPrestamo(tipo), tasaInteresAnual(tasaInteresAnual), cantidadCuotas(cantidadCuotas), montoPrestamo(montoPrestamo) {}

double Prestamo::calcularCuotaMensual() {
    double tasaInteresMensual = tasaInteresAnual / 12 / 100;
    double cuotaMensual = (montoPrestamo * tasaInteresMensual) / (1 - pow(1 + tasaInteresMensual, -cantidadCuotas));
    return cuotaMensual;
}
```

GENERAR TABLA



```cpp
// Se crea el objeto Prestamo con los datos ingresados
Prestamo prestamo(Prestamo::PERSONAL, tasaInteresAnual, cantidadCuotas, montoPrestamo);

// Se imprime el encabezado de la tabla en pantalla
std::cout << std::setw(8) << "Mes" << std::setw(15) << "Cuota Mensual" << std::setw(15) << "Intereses" << std::setw(15) << "Deuda" << s
std::cout << std::string(70, '-') << std::endl;

// Se calculan e imprimir los valores para cada mes
double deuda = montoPrestamo;
for (int i = 1; i <= cantidadCuotas; ++i) {
    double cuotaMensual = prestamo.calcularCuotaMensual();
    double intereses = deuda * (tasaInteresAnual / 12 / 100);
    double amortizacion = cuotaMensual - intereses;
    deuda -= amortizacion;
std::cout << std::setw(8) << i << std::fixed << std::setprecision(2) << std::setw(15) << cuotaMensual <<  std::setw(15) << intereses <<
}
cout << "operacion terminada, volviendo al menu principal"<< endl;
```