Simulador de Banco.
(26 de Noviembre, 2024)

Participantes: Johan Yesid Ramírez Torres, Néstor Iván Castellanos Merchán

Resumen: En el presente documento se realiza la respectiva documentación en Formato IEEE del punto E de laboratorio 2, el fin de este documento es presentar la metodología de dicho ejercicio.

I. INTRODUCCIÓN

En el presente formato encontraremos la solución del Punto E de laboratorio 2. El presente punto es referente a la simulación de un gestor de inventarios.

II. ANÁLISIS

Se tomarán en cuenta una serie de pasos y organización para la solución del ejercicio, es por ende de que como en la solución de este es poder hacer tramites bancarios sin necesidad de salir de casa

- Contexto: Se realiza el punto E de laboratorio que consiste en una cola de servicios y gestión virtual
- Población: Para usuarios que busquen hacer tramites bancarios de forma virtual
- Limitación y Alcance: Usuarios interesados en gestionar su cuenta bancaria desde casa

PROGRAMACION MODULAR

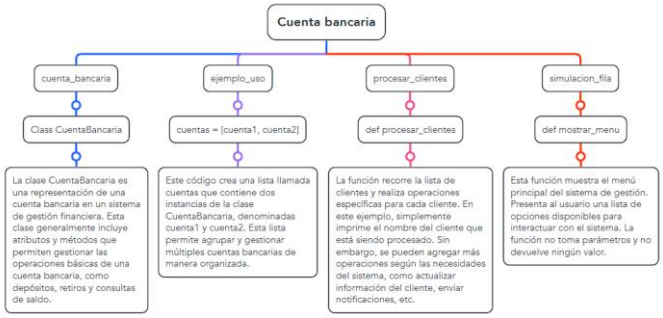

Figura 1: ProgramacionModular (Imagen fuente)

En la figura 1 se muestra la el mapa de programacion modular, la cual sera la estructura incial para trabajar sobre el ejercico E de laboratorio 2.

III. OBJETIVOS

- General:
  En el presente formato es dar información compleja sobre la realización de dicho ejercicio, e implementarlo y documentar respectivas funciones del código.

- Específicos:
  - Presentar un código preciso en la codificación totalmente funcional.
  - Enfatizar el uso respectivo de matrices y solución de errores en la codificación.
  - Dar en la metodología de problemas las fases completas.

IV. CODIFICACIÓN

A. Tipos de aplicación

En el presente software a realizar se tendrá en cuenta en la fase de Análisis de la Programación Modular y los conceptos completos respectivos a las Matrices para la verificación de valores respectivos.

Python:
En el presente lenguaje de programación “Python” se dará la realización del Punto E de laboratorio 2, se utilizarán funciones y matrices para su respectiva solución.

```python
class CuentaBancaria:
    def __init__(self, numero_cuenta, saldo_inicial=0):
        self.numero_cuenta = numero_cuenta
        self.saldo = saldo_inicial

    def depositar(self, monto):
        if monto > 0:
            self.saldo += monto
            print(f"Depósito de {monto} realizado. Nuevo saldo: {self.saldo}")
        else:
            print("El monto del depósito debe ser positivo.")

    def retirar(self, monto):
        if 0 < monto <= self.saldo:
            self.saldo -= monto
            print(f"Retiro de {monto} realizado. Nuevo saldo: {self.saldo}")
        else:
            print("Fondos insuficientes o monto inválido.")

    def consultar_saldo(self):
        print(f"El saldo de la cuenta {self.numero_cuenta} es: {self.saldo}")
        return self.saldo

```

Figura 2: Codificacion_Python1 (Imagen fuente)

- Este código define una clase CuentaBancaria que se utiliza para gestionar operaciones básicas de una cuenta bancaria. La clase incluye métodos para depositar dinero, retirar dinero y consultar el saldo de la cuenta.

```python
from cuenta_bancaria import CuentaBancaria

# Ejemplo de uso
cuenta1 = CuentaBancaria("001", 1000)
cuenta2 = CuentaBancaria("002", 500)

cuenta1.depositar(200)
cuenta1.retirar(150)
cuenta1.consultar_saldo()

cuenta2.depositar(300)
cuenta2.retirar(100)
cuenta2.consultar_saldo()

# Exportar las cuentas para su uso en otras partes del programa
cuentas = [cuenta1, cuenta2]

```

Figura 3: Codificacion_Python2 (Imagen fuente)

- Este código implementa un sistema de gestión de cuentas bancarias utilizando la clase CuentaBancaria. Se crean dos instancias de CuentaBancaria con números de cuenta y saldos iniciales específicos.

```python
from cuenta_bancaria import CuentaBancaria

def procesar_clientes(fila_clientes):
    while fila_clientes:
        cliente = fila_clientes.pop(0)
        print(f"Atendiendo al cliente con cuenta número: {cliente.numero_cuenta}")
        cliente.consultar_saldo()

```

Figura4: Codificacion_Python4 (Imagen fuente)

- Este código implementa un sistema básico para procesar clientes en una fila de un banco utilizando la clase CuentaBancaria. Primero, se importa la clase CuentaBancaria desde el módulo correspondiente. Luego, se define la función procesar_clientes, que toma una lista de clientes (fila_clientes) y los procesa uno por uno.

```python
from ejemplo_uso import cuentas

def mostrar_menu():
    print("\nBienvenido al Banco")
    print("1. Seleccionar cuenta")
    print("2. Depositar")
    print("3. Retirar")
    print("4. Consultar saldo")
    print("5. Salir")

def seleccionar_cuenta():
    print("\nSeleccione una cuenta:")
    for i, cuenta in enumerate(cuentas):
        print(f"{i + 1}. Cuenta {cuenta.numero_cuenta}")
    seleccion = int(input("Ingrese el número de la cuenta: ")) - 1
    return cuentas[seleccion]

def realizar_operacion(cuenta):
    while True:
        mostrar_menu()
        opcion = int(input("Seleccione una opción: "))
        if opcion == 1:
            cuenta = seleccionar_cuenta()
        elif opcion == 2:
            monto = float(input("Ingrese el monto a depositar: "))
            cuenta.depositar(monto)
        elif opcion == 3:
            monto = float(input("Ingrese el monto a retirar: "))
            cuenta.retirar(monto)
        elif opcion == 4:
            cuenta.consultar_saldo()
        elif opcion == 5:
            print("Gracias por usar el Banco. ¡Hasta luego!")
            break
        else:
            print("Opción no válida. Intente de nuevo.")

# Iniciar la interfaz de usuario
cuenta_seleccionada = seleccionar_cuenta()
realizar_operacion(cuenta_seleccionada)

```

Figura5: Codificacion_Python4 (Imagen fuente)

- Este código implementa un sistema de gestión bancaria que permite al usuario realizar diversas operaciones sobre cuentas bancarias. Primero, se importan las cuentas desde el módulo ejemplo_uso. Luego, se define la función mostrar_menu que presenta al usuario un menú con opciones para seleccionar una cuenta, depositar, retirar, consultar saldo y salir del sistema. La función

seleccionar_cuenta permite al usuario elegir una cuenta de la lista de cuentas disponibles. La función realizar_operacion gestiona las operaciones bancarias, mostrando el menú, leyendo la opción seleccionada por el usuario y ejecutando la operación correspondiente. Finalmente, se inicia la interfaz de usuario seleccionando una cuenta y llamando a la función realizar_operacion. Este código es útil para simular operaciones bancarias básicas y gestionar cuentas de manera eficiente en un sistema financiero.

- Requerimientos del sistema

- Para teléfono:
  Android: Versión 10
  RAM: 2GB
  Almacenamiento: 500Mb

- Para Ordenador:
  Windows 7 o superior
  RAM: 4GB
  Almacenamiento: 500Mb

V. Diseño del algoritmo

En el presente punto se da a conocer el diseño de algoritmo teniendo presente su proceso y explicación, en concreto con la programación modular donde esta misma es sutil en cuanto la estructura en la codificación.

VI. Ejecución del programa

En la correspondiente ejecución del programa se establece sin novedad su funcionamiento como tal, siendo este logrando conseguir que se logre de antemano su finalidad bajo los parámetros y conclusión de errores.

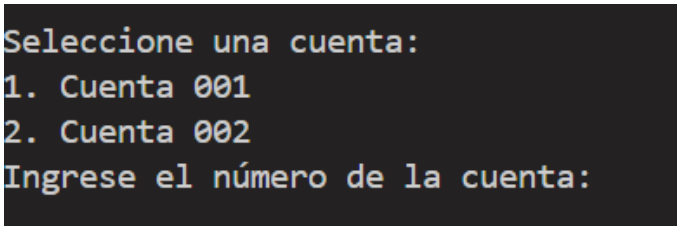

Figura 6: EjecucionPython1 (Imagen fuente)

- En la figura 6 encontramos la ejecución del programa. Se muestra el menú del usuario con 2 opciones
- Cuenta 001
- Cuenta 002

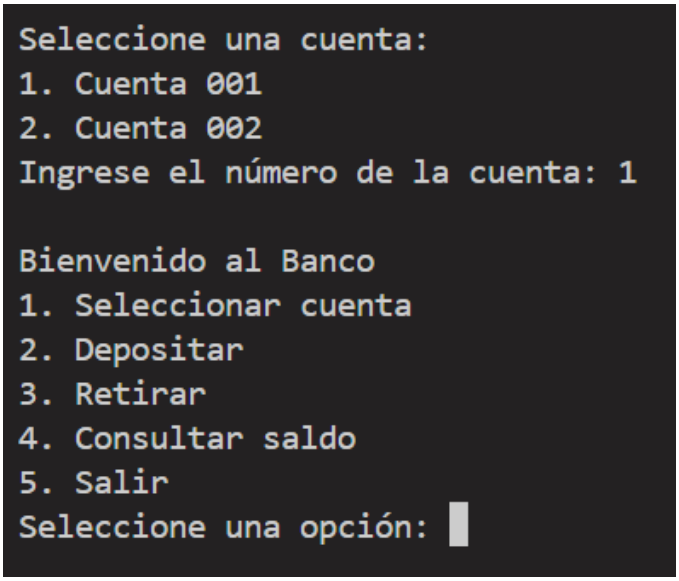

Figura 7: EjecucionPython2 (Imagen fuente)

- En la figura 7 encontramos la ejecución del menú numero 1 la cual su función es elegir la cuenta de donde queremos ingresar al banco

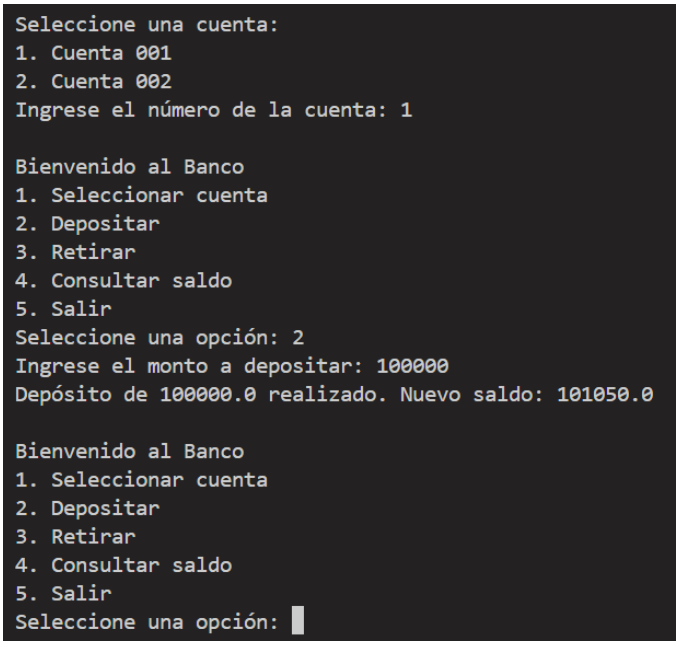

Figura 8: EjecucionPython3 (Imagen fuente)

- En la figura 8 encontramos en funcionalidad la opción del menú numero 2 la cual muestra la opción de realizar un depósito a nuestra cuenta

```
Seleccione una cuenta:
1. Cuenta 001
2. Cuenta 002
Ingrese el número de la cuenta: 1

Bienvenido al Banco
1. Seleccionar cuenta
2. Depositar
3. Retirar
4. Consultar saldo
5. Salir
Seleccione una opción: 3
Ingrese el monto a retirar: 50000
Retiro de 50000.0 realizado. Nuevo saldo: 51050.0

Bienvenido al Banco
1. Seleccionar cuenta
2. Depositar
3. Retirar
4. Consultar saldo
5. Salir
Seleccione una opción:
```

Figura 9: EjecucionPython4 (Imagen fuente)

- En la figura 9 encontramos la opción 3 para realizar un retiro de la cuenta bancaria

```
Bienvenido al Banco
1. Seleccionar cuenta
2. Depositar
3. Retirar
4. Consultar saldo
5. Salir
Seleccione una opción: 4
El saldo de la cuenta 001 es: 51050.0
```

Figura 10: EjecucionPython4 (Imagen fuente)

- En la figura 10 nos muestra la opción 4 para consultar el saldo disponible de la cuenta

```
Bienvenido al Banco
1. Seleccionar cuenta
2. Depositar
3. Retirar
4. Consultar saldo
5. Salir
Seleccione una opción: 5
Gracias por usar el Banco. ¡Hasta luego!
```

Figura 10: EjecucionPython4 (Imagen fuente)

- En la figura 11 nos muestra la opción final, la cual funciona para cerrar la operacion

VII. CONCLUSIÓN

- En la respectiva documentación se da por finalizado el formato IEEE referente al punto E de laboratorio 2, respectiva documentación se encuentra la programación modular y codificación completa dada en el lenguaje de programación de Python.