Proyecto #6

Grupo 6

Lógica Computacional

7:00 - 9:00 Am Martes y Jueves

- Evelin Solano
- Laura Fonseca
- Daniela Portela

Actividades Preliminares

9). Resuma el ciclo de vida de construcción de un programa.

El ciclo de vida de un programa se compone de varias fases, entre ellas, la planificación (define las necesidades del usuario, y establece el alcance del proyecto); el diseño (funcionalidades del programa); el desarrollo (estructura código); pruebas (se verifica que el código actue bien); y mantenimiento (se mantiene funcional el software).

b). Explique los aspectos que hacen parte del análisis de un programa.

- Naturaleza: Comprender el tipo de problema, su alcance y el impacto que tiene
- Causas: Identificar las causas que originan el problema, tanto principales como secundarias.
- Consecuencias: Determinar las consecuencias del problema y como afecta a las personas
- Importancia: Evaluar la importancia del problema para la comunidad y para las personas afectadas.
- Frecuencia: Considerar con que frecuencia ocurre.
- Magnitud: Saber cuantas personas se ven afectadas.
- Gravedad: Evaluar si el problema afecta la calidad de vida de las personas.
- Prevención: Analizar si es posible revertir los efectos negativos del problema.

C. Explique las etapas del proceso de solución de problemas.

Nivel 1. Problemas, Soluciones y Programas.

- Identificar y entender el problemc
- Validar los datos que se conocen
- Establecer requisitos y restricciones del programa.

Nivel 2 Manejo de casos

- Estrategia para resolver el problema
- Dividir el programa en subtemas manejables
- Crear un algoritmo que describe paso a paso
- Escoger las herramientas correctas

Nivel 3. Implementación.

- escribir código
- programa estructurado y fácil de entender

Nivel 4: Pruebas y depuración.

- probar programa con diferentes escenarios
- Identificar y corregir fallos o problemas.

Nivel 5: Evaluación.

- Analizar si la solución cumple con los objetivos
- Verificar si se puede optimizar el programa

Nivel 6: Documentación y mantenimiento

- Documentar el código que sea fácil interpretar
- Estar preparado para actualización de código a futuro

d). ¿Cuáles son los elementos que se deben entregar a un cliente?

Primeramente debemos identificar el problema, entender, las necesidades del cliente, y ofrecer soluciones.

Al momento de ofrecer la solución al cliente, debemos ofrecer la solución más atractiva, y variar opciones entre más sea posible, mostrando los beneficios y ventajas de productos o servicios y respaldar los con datos y evidencias.

Estas soluciones deben responder a sus objeciones y dudas con argumentos sólidas y convincentes, demostrando ser la mejor opción para satisfacer sus necesidades.

F. Elabore la tarea No. 2 (pag 13), con el objetivo de identificar los requerimientos funcionales de un problema.

En el CD que acompaña el libro, puede encontrar el formulario que debe llenar un programador para especificar los requerimientos funcionales de un pragrama.

Objetivo crear habilidad en la identificación y especificación de requerimientos funcionales. para el caso de estudio 2, un simulador bancario, identifique y especifique tres requerimientos funcionales.

Requerimiento funcional 1	Nombre	Realizar deposito en cuenta
	Resumen	Permitir que un usuario deposite fondos en su cuenta bancaria, aumentando el saldo de la misma
	Entradas	Número de cuenta - monto a depositar
	Resultado	El saldo de la cuenta incrementa por el monto depositado
Requerimiento funcional 2	Nombre	Retirar fondos de la cuenta.
	Resumen	Permitir que un usuario retire fondos de su cuenta bancaria disminuyendo el saldo de la misma
	Entradas	Número de cuenta - monto a retirar
	Resultado	El saldo de la cuenta se reduce por el monto retirado, siempre que haya fondos suficientes

Requerimiento funcional 3.	Nombre	Consultar saldo
	Resumen	Mostrar al usuario el saldo actual de su cuenta bancaria
	Entrados	Número de cuenta
	Resultados	Se presenta el saldo disponible al usuario.

6. Elabore la tarea No. 3 (pag 14) con el objetivo de identificar los requerimientos funcionales de un problema.

Objetivo: Crear habilidad en la identificación y especialización de requerimientos funcionales.

Para el caso de estudio 3, un programa para manejar un triángulo, identifique y especifique tres requerimientos funcionales.

Requerimiento funcional 1	Nombre	Determine el área de un triángulo
	Resumen	El programa debe calcular el área de un triángulo basado en su base y altura.
	Entradas	Base triangulo - altura triángulo
	Resultados	El área de triángulo calculada, usando Área = $\frac{\text{base} \times \text{Altura}}{2}$
Requerimiento funcional 2	Nombre	Determinar el tipo de triángulo
	Resumen	El programa debe clasificar el triangulo según sus lados (equilatero isosceles o escaleno)
	Entradas	Longitudes de los tres lados del triángulo
	Resultados	Una etiqueta que indica el tipo de triángulo

Requerimiento funcional 3	Nombre	Verificar si es un triángulo válido
	Resumen	El programa debe determinar si los tres lados forman un triángulo usando la desigualdad triangular
	Entrados	Longitudes de los tres lados del triángulo
	Resultados	Una confirmación de si las medidas forman un triángulo válido o no.

Punto H.

Tarea #4.

Punto de reflexión.

1. ¿Qué pasa si no identificamos bien las entidades del mundo?

→ Una identificación errónea podría llevar a problemas que afecten la comprensión diseño y mantenimiento del programa. Ya que una interpretación correcta ayuda al buen desarrollo del programa.

2. ¿Cómo decidir si se trata efectivamente de una entidad y no solo de una característica de una entidad ya identificada?

→ Analizando la funcionalidad y relaciones con otro elemento del problema. Las características nos suelen detallar las entidades

	Nombre	Descripción
Entidad	CuentaBancaria	Es la entidad más importante, puesto que define su frontera. Es buena práctica comenzar la etapa de análisis tratando de identificar la clase más importante del problema. Cuando el nombre de la entidad es compuesto, se usa por convención una letra mayúscula al comienzo de la palabra.
Entidad	CuentaAhorros	Este también incluye una cuenta de ahorros. Los nombres asignados a las clases deben ser significativos y dar una idea clara de la entidad del mundo que representan. No se debe exagerar con la longitud del nombre.
Entidad	Mes	Clase que nos dirá en cuál mes de la simulación se encuentra la cuenta Bancaria La simulación se hace mes a mes y existe un requerimiento funcional que permite avanzar un mes la simulación.

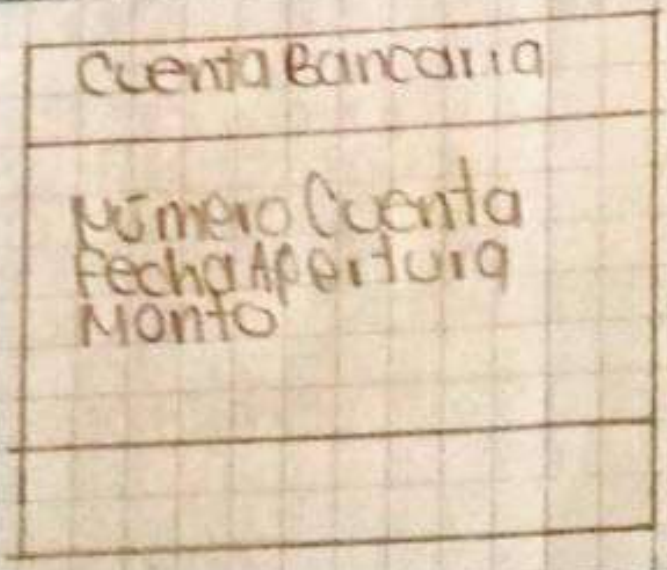
Punto I

Tarea #5.

Clase: CuentaBancaria

Atributo	Valores posibles
Número Cuenta	20 digitos
Fecha apertura	valores enteros positivos
Monto	valores enteros positivos

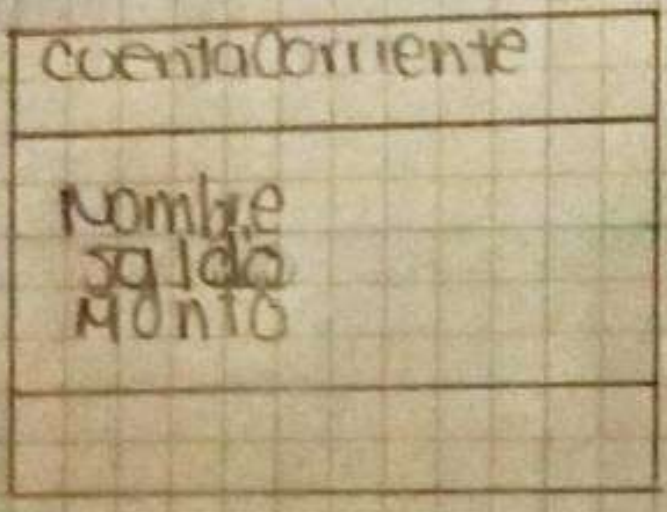
Diagrama UML



Clase: CuentaCorriente.

Atributo	Valores posibles
Nombre	Cadena de caracteres
Saldo	Cualquier valor.
Monto	valores enteros positivos

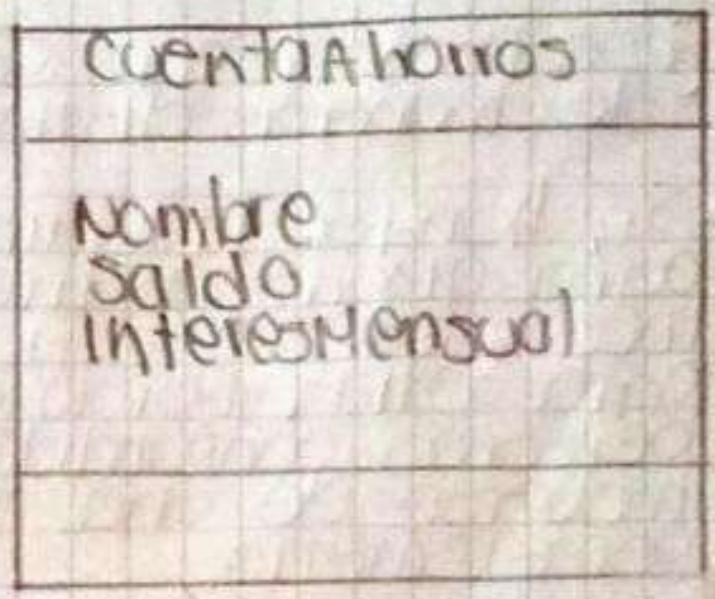
Diagrama UML



clase: CuentaAhorros

Atributo	Valores posibles
Nombre	Cadena de caracteres
saldo	Cualquier valor
Interes, mensual	una parte del dinero en la cuenta

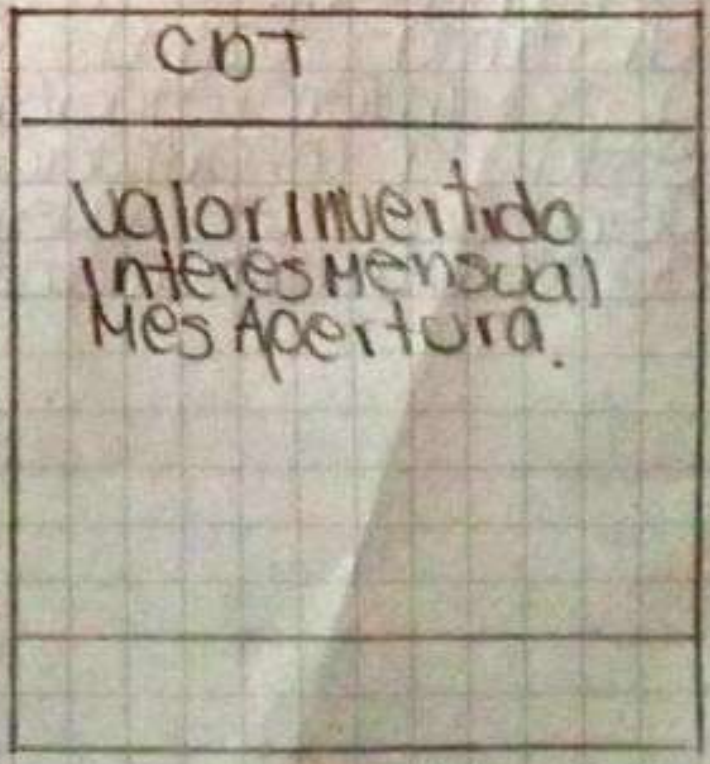
Diagrama UML



Clase = CDT

Atributo	Valores posibles
Valor, invertido	Valores positivos
Interes, mensual	una parte del dinero en la cuenta
mes, apertura	1 y 12

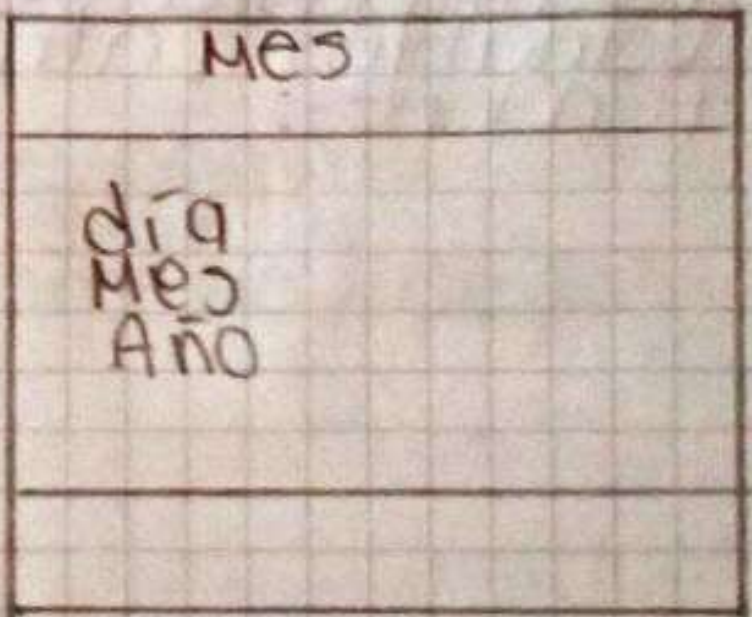
Diagrama UML



Clase: Mes

Atributo	valores posibles
día	1 y 31
Mes	1 y 12
Año	valores enteros positivos

Diagrama UML



Punto j.

Tarea #6.

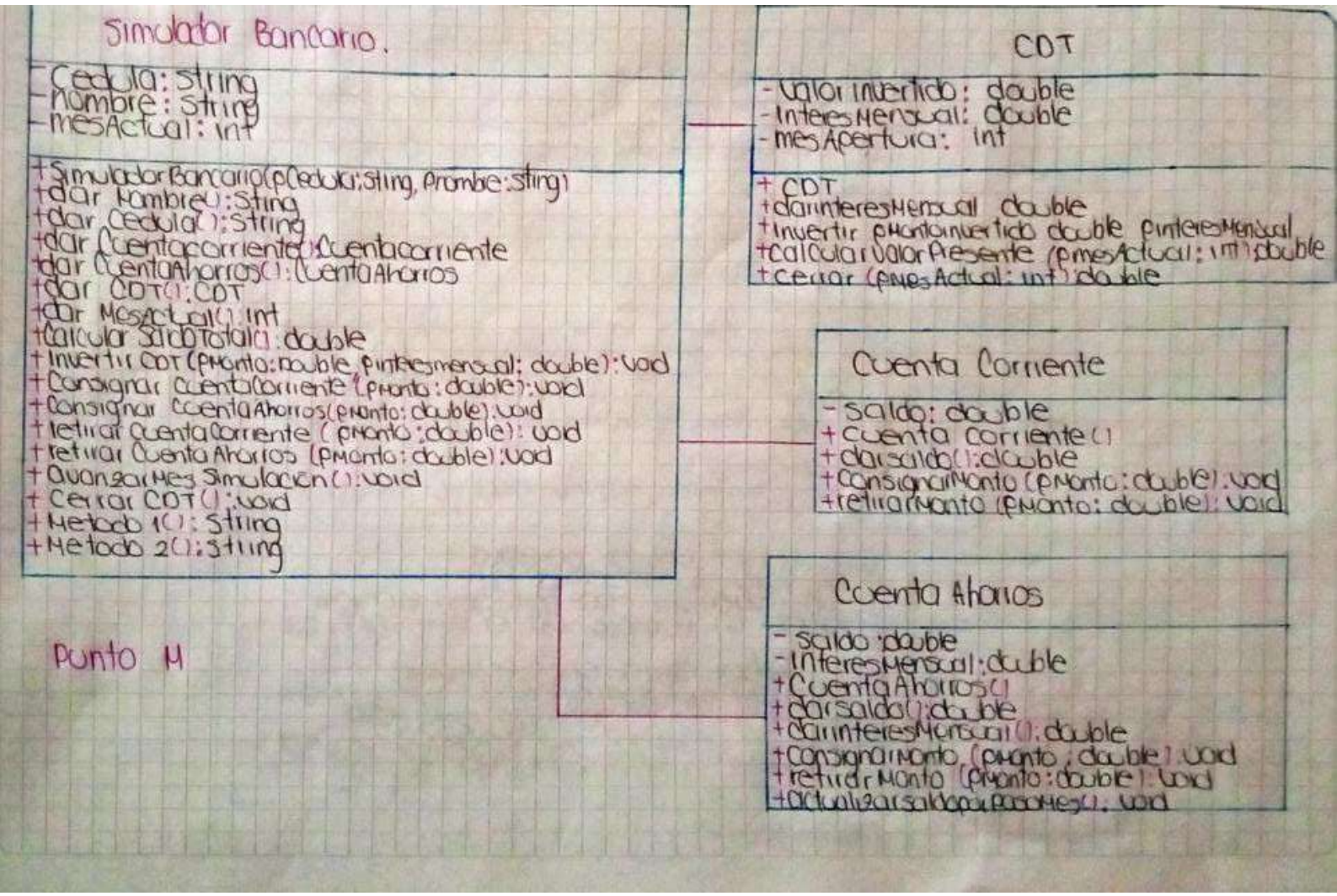
Conclusiones.

¿Se prestan para que se interpreten de maneras distintas?

→ Sí, aunque la información estaría clara para algunos, podría ser considerada incompleta o difícil de entender para otras. Por lo que, las personas confundidas podrían moverse de una manera errónea por el metro.

¿estamos suponiendo que quien lo lee usa su "sentido común", o cualquier persona que lo use va a resolver siempre el problema de la misma manera?

→ definitivamente cierto grupo de personas va a resolverlo de distintas formas, ya que, lo harían a su interpretación, aunque, al estar la información clara, muchas personas resolverían la situación de la misma manera.


Simulador Bancario.
-Cedula: String
-nombre: String
-mesActual: int
+SimuladorBancario(pCedula:String, pNombre:String)
+dar Nombre():String
+dar Cedula():String
+dar CuentaCorriente():CuentaCorriente
+dar CuentaAhorros():CuentaAhorros
+dar CDT():CDT
+dar MesActual():int
+Calcular SaldoTotal():double
+Invertir CDT(pMonto:double, pInteresMensual: double):void
+Consignar CuentaCorriente(pMonto:double):void
+Consignar CuentaAhorros(pMonto:double):void
+Retirar CuentaCorriente(pMonto:double):void
+retirar CuentaAhorros(pMonto:double):void
+AvanzarMesSimulacion():void
+Cerrar CDT():void
+Metodo1():String
+Metodo2():String
CDT
-ValorInvertido: double
-InteresMensual: double
-mesApertura: int
+CDT
+darInteresMensual double
+Invertir pMontoInvertido double pInteresMensual
+CalcularValorPresente(pmesActual:int):double
+cerrar(pMesActual:int):double
Cuenta Corriente
-saldo: double
+Cuenta Corriente()
+darSaldo():double
+ConsignarMonto(pMonto:double):void
+retirarMonto(pMonto:double):void
Cuenta Ahorros
-saldo:double
-InteresMensual:double
+CuentaAhorros()
+darSaldo():double
+darInteresMensual():double
+ConsignarMonto(pMonto:double):void
+retirarMonto(pMonto:double):void
+actualizarSaldoporPeso():void

Punto M

Acción

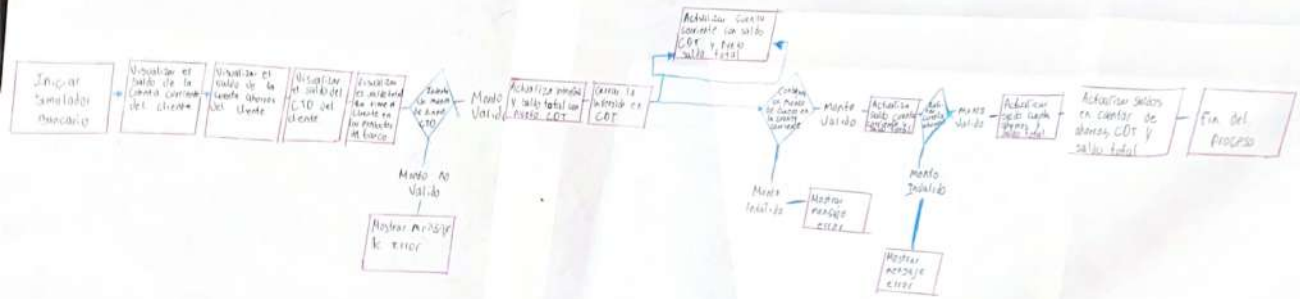
K). Estudia los siguientes aspectos del ejemplo seleccionado: Enunciado, Requerimientos funcionales (casos de uso) y el modelo (clases de proyecto). A continuación redacta el enunciado del problema y el nombre de cada uno de los requerimientos funcionales del proyecto.

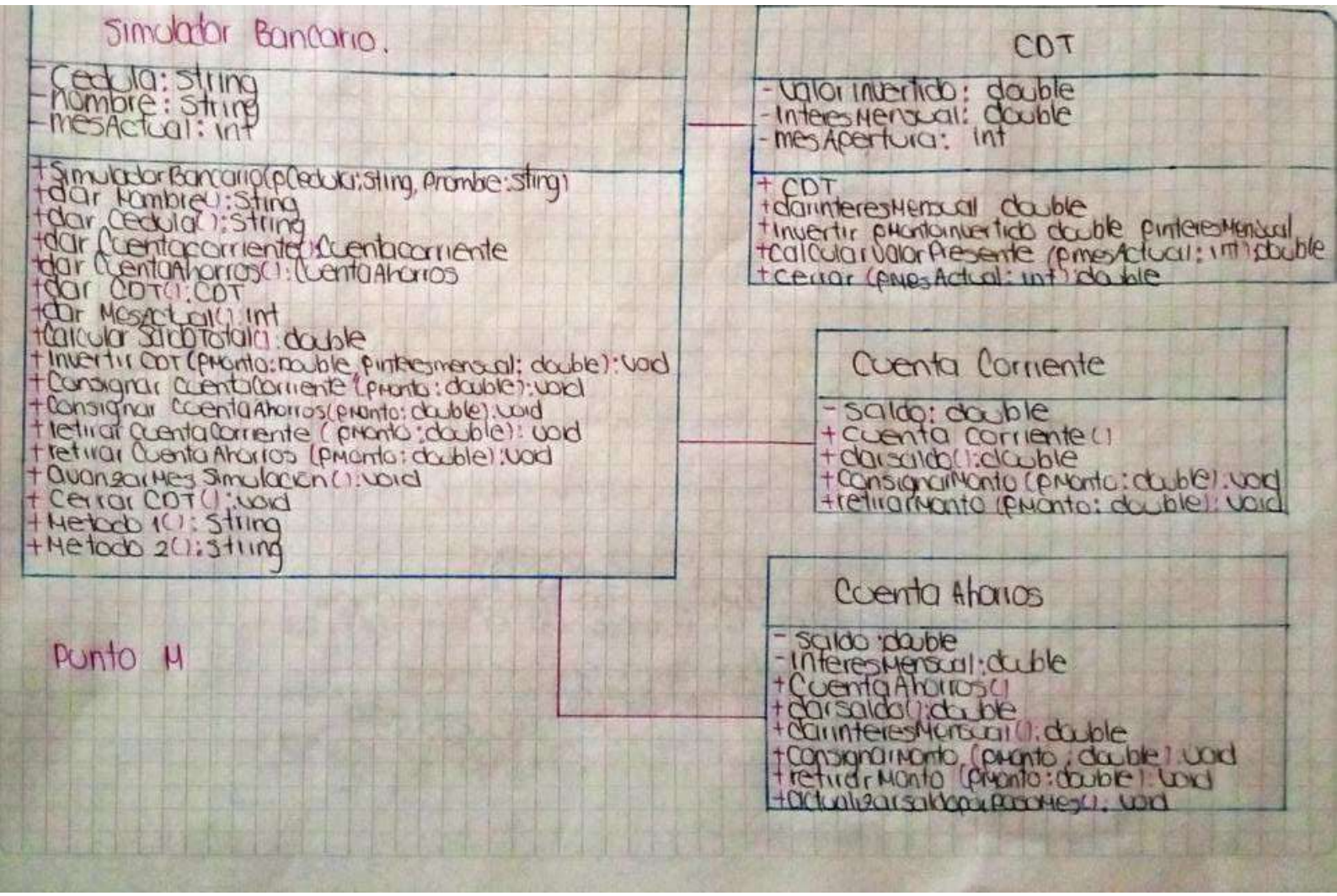
Se quiere crear un programa que permita simular el comportamiento de las cuentas bancarias de un cliente, estas son la cuenta corriente (solo retirar y depositar dinero, sin intereses), una cuenta de ahorros (depositar o retirar dinero, recibe 0.6% de interes mensual sobre el saldo actual), Un CDT (no consigna ni retira dinero, solamente se puede cerrar y en este caso el dinero e intereses pasan a la cuenta corriente. El saldo total es la suma de con lo que tiene cada cuenta del cliente.

- Visualizar saldo de la cuenta corriente del cliente
- Visualizar saldo de la cuenta de ahorros del cliente
- Visualizar saldo del CDT del cliente
- Visualizar el saldo total que tiene el cliente en los productos del banco.

- Invertir un monto de dinero en un CDT.
- Cerrar la inversión en CDT
- Consignar un monto de dinero a la cuenta corriente
- Retirar un monto de la cuenta corriente.
- Consignar un monto a la cuenta de ahorros.
- Retirar un monto de la cuenta de ahorros.
- Avanzar en un mes la simulación.

1. Dibuja el respectivo diagrama de casos de uso del ejemplo Simulador Bancario




Simulador Bancario.
-Cedula: String
-nombre: String
-mesActual: int
+SimuladorBancario(pCedula:String, pNombre:String)
+dar Nombre():String
+dar Cedula():String
+dar CuentaCorriente():CuentaCorriente
+dar CuentaAhorros():CuentaAhorros
+dar CDT():CDT
+dar MesActual():int
+Calcular SaldoTotal():double
+Invertir CDT(pMonto:double, pInteresMensual: double):void
+Consignar CuentaCorriente(pMonto:double):void
+Consignar CuentaAhorros(pMonto:double):void
+Retirar CuentaCorriente(pMonto:double):void
+retirar CuentaAhorros(pMonto:double):void
+AvanzarMesSimulacion():void
+Cerrar CDT():void
+Metodo1():String
+Metodo2():String
CDT
-ValorInvertido: double
-InteresMensual: double
-mesApertura: int
+CDT
+darInteresMensual double
+Invertir pMontoInvertido double pInteresMensual
+CalcularValorPresente(pmesActual:int):double
+cerrar(pMesActual:int):double
Cuenta Corriente
-saldo: double
+Cuenta Corriente()
+darSaldo():double
+ConsignarMonto(pMonto:double):void
+retirarMonto(pMonto:double):void
Cuenta Ahorros
-saldo:double
-InteresMensual:double
+CuentaAhorros()
+darSaldo():double
+darInteresMensual():double
+ConsignarMonto(pMonto:double):void
+retirarMonto(pMonto:double):void
+actualizarSaldoporPeso():void

Punto M

Punto N

Requerimiento funcional 1	Nombre	Gestión de notas
	Resumen	El programa debe permitir ingresar almacenar y calcular el promedio de los estudiantes de un curso
	Entradas	Nombre del estudiante Numero de notas por estudiante Calificación de cada estudiante
	Resultado	promedio de estudiante estudiante con nota más alta promedio de curso

Requerimiento funcional 2	Nombre	Reservas de Vuelo.
	Resumen	El programa debe de gestionar las reservas de asientos se podrá escoger asientos disponibles verificar estado de vuelo y cancelarlo
	Entradas	Numero de asientos solicitud de reserva cancelación de vuelo
	Resultados	Asientos ocupados y disponibles Porcentaje de ocupación Confirmación de reservas