Ejemplo de herencia.

Un banco almacena la información de todas las cuentas bancarias de los clientes que operan en el mismo.

a) Crear el banco Del Chubut y agregarle varias cuentas (cajas de ahorro y cuentas corrientes)
b) Simular la atención de varios clientes que van a depositar y/o extraer.
c) Se desea sabe cuál es el monto total que dispone el banco en calidad de dinero depositado por los clientes en sus cuentas respectivas.
d) Construir un diccionario que informe para cada cliente la cantidad total de cuentas que posee en el banco.

Diseño de clases usando herencia.

Clase Banco, subclase de Object
v.i. nom conjCuentas

M.clase
>>crearBanco: unnombre
^super new init: unnombre

M.instancia
>>init: unN
nom:=unN.
conjCuentas:=OrderedCollectin new.

>>agregaCuenta: unaC
"agrega una cuenta al conjunto de cuentas, ya sea caja de ahorro o cuenta corriente, quedan todas juntas y mezcladas"
conjCuentas add:unaC.

>>eliminaCuenta: unaC
conjCuentas remove:unaC.

>>cantidadCuentas
^conjCuentas size.

>>todasCuentas
^conjCuentas

>> existeCuenta: unaC
^conjCuentas includes:unaC

>>buscarCuenta: unNro
^conjCuentas detect:[: ele | ele vernNro = unNro] ifNone :[nil].

>>retornaCuenta: pos
^conjCuentas at: pos.

verNom
^nom

modNom:otro
nom:otroN.

Cuenta Bancaria, subclase de Object
v.i. “nroCuenta titular saldo”

M.Clase
>>crearCB:nro con:titu
^ super new iniciarCB: nro con: titu

M.instancia
>>iniciarCB: num con: tit
nroCuenta:=num.
titular:=tit.
saldo:=0.

>>verSaldo
^saldo

>>verTit
^titular

>>modTit:otro
titular:=otro.

>>verNro
^nroCuenta

>>modNro:otro
nrocuenta:=otro.

>>depositar:unMonto
saldo:=saldo + unMonto.

>>extraer: unMonto
^ ((self verifCtaHab) & (self verifOpVálida)) ifTrue:[self realizarOp:unMonto]. “usa herencia con el self”

Clase CajaAhorro, subclase de CuentaBancaria
v.i. “cantExtracciones”

M.Clase
>>crearCA: num con: titu

^ super new iniciaCA:num con:titu "se puede usar self new o super new"

Si usamos super new, busca desde CuentaBancaria.

M.instancia
>>iniciaCA: nro con:tit
"inicializa la caja de ahorro con todos sus datos , se programa usando herencia"
super iniciarCB:nro con:tit. "utiliza herencia invocando al método de la super clase"
cantExtracciones:=0.

>>verCanExt
^cantExtracciones

>>incExt
cantExtracciones:=cantExtracciones + 1

>>verifCtaHab
(cantExtracciones < 5) ifTrue :[^ true]
ifFalse :[^false].

>>verifOpVálida: unMonto
(saldo >= unMonto) ifTrue :[^true]
ifFalse:[^false].

>>realizarOp:unMonto
saldo:=saldo – unMonto.
self incExt .

Clase CuentaCte, subclase de CuentaBancaria
v.i. " saldoEnRojo cheque"

M.Clase
>>crearCC: num con: titu con: enRojo
^ super new iniciaCC:num con:titu con:enRojo

M.instancia
>>iniciaCC: nro con:tit con: enRojo
"se inicializa la cuenta corriente con todos sus datos, se programa usando herencia"
super iniciarCB:nro con:tit.
saldoEnRojo:=enRojo.
cheque:="no".

>>verSaldoRojo
^saldoEnRojo

>>modSaldoRojo:otro
saldoEnrojo:=otro.

>>verCheque
^cheque

```
>>modCheque: val
cheque:=val.

>> verifCtaHab
        (cheque =“no”)  ifTrue :[^true]
                        ifFalse:[^false].

>> verifOpVálida: unMonto
(unMonto <= (saldo + saldoRojo)) ifTrue :[^true]
                                 ifFalse:[^false].

>>realizarOp:unMonto
saldo:=saldo - unMonto.
```

Aplicación

```
ban:=Banco crearBanco:’del Chubut’.
resp:=Prompter prompt: ‘Desea ingresar una cuenta?s/n’.
[resp=P=’s’]  whileTrue: [  num:=Prompter prompt: ‘ingrese numero de cuenta’.
                            titu:=Prompter prompt: ‘ingrese el dni del titular  de cuenta’.

                            tipo:=Prompter prompt: ‘Ingrese 1 si va a crear una Caja de ahorro- 2 si es una cuenta corriente’.
                           (tipo=1) ifTrue:[  cuen:=CajaAhorro crearCA: num con:titu.]
                                    ifFalse:[ sal:=Prompter prompt: ‘ingrese monto del saldo en rojo’.
                                              cuen:=CuentaCte crearCC:num con: titu con: sal.].

                           ban agregarCuenta: cuen.   “agrega una caja de ahorro o una cuenta cte dependiendo de lo que se haya creado en el paso previo”

                           resp:=Prompter prompt: ‘Desea ingresar otra cuenta?s/n’.
                ].


“resuelvo inciso b”
resp:=Prompter prompt: ‘Desea hacer una operación?s/n’.
[resp=P=’s’]  whileTrue: [  num:=Prompter prompt: ‘ingrese numero de cuenta’.
                            cuen:=ban buscarCuenta: num.
                           (cuen notNil) ifTrue:[
                           tipo:=Prompter prompt: ‘Ingrese 1 si va a depositar- 2 si es una extracción’.
                           monto:=Prompter prompt: ‘ingrese el monto de la operación’.
                           (tipo=1) ifTrue:[  cuen depositar: monto.]
                                    ifFalse:[ cuen extraer: monto.].  “depende de la clase a la que pertenezca cuen usa los métodos correspondiente”

                           resp:=Prompter prompt: ‘Desea ingresar otra operación?s/n’.
                ].
```

```
“resuelvo el inciso c”
col:= ban todasCuentas.
suma:=0.
col do: [: c| suma:= suma +  c verSaldo].

“resuelvo el inciso d”
diccio:=Dictionary new.
colTitulares:= col collect:[: el|  el verTitu].
sinRep:=colTitulares asSet.
sinRep do: [ : nro|    cant:= colTitulares occurrencesOf: nro.
                       diccio at: nro put: cant].
diccio keysdo: [: cla|   Transcript show:  cla, ‘     ‘, (( diccio at:cla) displayString)].
```