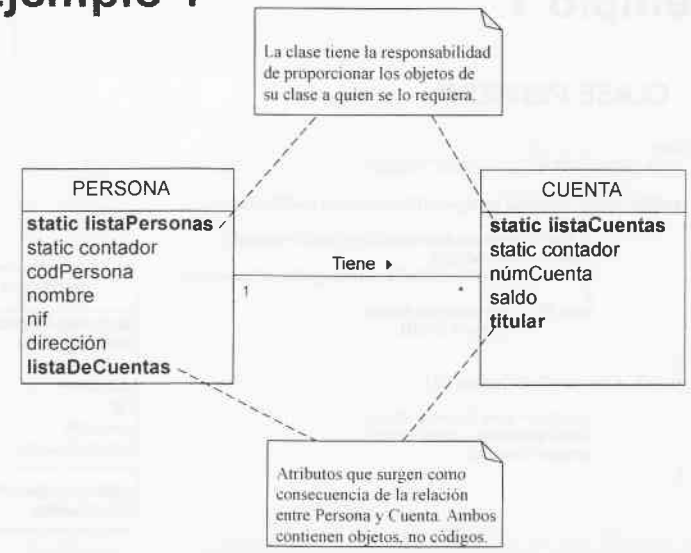
Ejemplo 1

Gestión de cuentas bancarias

Ejemplo 1

- Clases implicadas
  - Persona
  - Cuenta

Ejemplo 1



Ejemplo 1

CLASE PERSONA

```
class Persona {
        private static Hashmap<Integer, Persona> listaPersonas =
                new Hashmap<Integer, Persona>();
        private static int contador = 0;
        private int codPersona;
        private String nombre;
        private String nif;
        private String direccion;
        private ArrayList<Cuenta> listaDeCuentas =
                new ArrayList<Cuenta>();

        Persona ( String nombrePersona, String niffPersona,
                String direccionPersona) {
                contador++;
                codPersona = contador;
                nombre = nombrePersona;
                nif = niffPersona;
                direccion = direccionPersona;
                listaPersonas.put(codPersona, this);
        }
}
```

Es MUY importante determinar el tipo correcto de lista.

PERSONA
static listaPersonas
static contador
codPersona
nombre
nif
dirección
listaDeCuentas

A partir de la versión 1.5 de Java, se permite usar datos elementales en lugar de sus clases envoltorias. Esto se conoce como Autoboxing.

Ejemplo 1

CLASE CUENTA

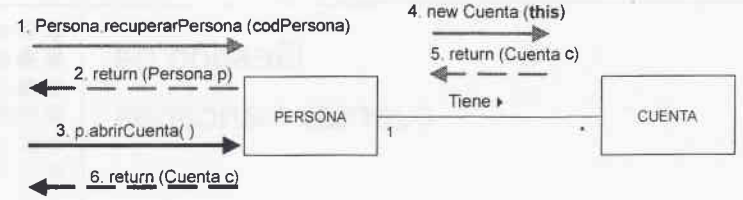
No puede haber una cuenta sin titular, de ahí que se asigne ese atributo en el constructor. En cambio, una persona puede estar dada de alta sin tener asociada una cuenta, de ahí que no se haya inicializado ese atributo en su constructor.

```
class Cuenta {
        private static Hashmap<Integer, Cuenta> listaCuentas = new Hashmap<Integer, Cuenta>();
        private static int contador = 0;
        prvate int numCuenta;
        private float saldo;
        private Persona titular;

        Cuenta (Persona persona) {
                contador++;
                numCuenta = contador;
                saldo = 0;
                titular = persona;
                listaCuentas.put(numCuenta, this);
        }
        Cuenta (Persona persona, float saldoInicial) {
                contador++;
                numCuenta = contador;
                titular = persona;
                saldo = saldoInicial;
                listaCuentas.put(numCuenta, this);
        }
}
```

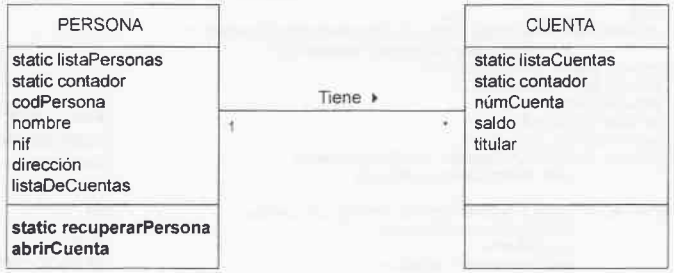
CUENTA
static listaCuentas static contador **númCuenta** **saldo** **titular**

Ejemplo 1

- Abrir cuenta



Ejemplo 1



Ejemplo 1

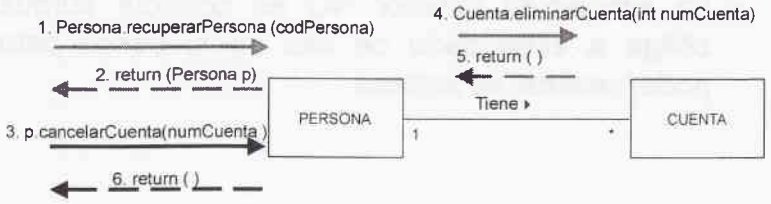
CLASE PERSONA

```
class Persona {
        //ATRIBUTOS Y CONSTRUCTORES

        public static Persona recuperarPersona (int codPersona) {
                ...
                if (listaPersonas.containsKey(codPersona)) {
                //Esa persona existe
                        return (listaPersonas.get(codPersona));
                }
                else //Esa persona no existe
                        return (null);

        }
        public Cuenta abrirCuenta ( ) {
                ...
                cuenta = new Cuenta (this);
                listaDecuentas.add(cuenta);
                return (cuenta);
        }
}
```

PERSONA
static listaPersonas static contador codPersona nombre nif dirección listaDeCuentas
static recuperarPersona **abrirCuenta**

Ejemplo 1

- Cancelar cuenta

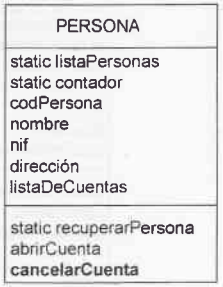


Ejemplo 1

Ejemplo 1

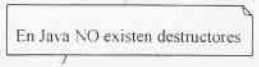
CLASE PERSONA

```
class Persona {
        //ATRIBUTOS Y CONSTRUCTORES
        //RESTO DE MÉTODOS
        public boolean cancelarCuenta (int numCuenta ) {
                ...
                // buscar la cuenta en listaDeCuentas
                // Si no está, devolver Falso
                // sino...
                Cuenta.eliminarCuenta(numCuenta));
                // Quitar la cuenta de listaDeCuentas
                // y devolver Verdadero
        }
}
```

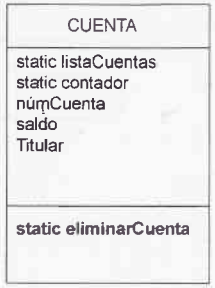


Ejemplo 1

CLASE CUENTA

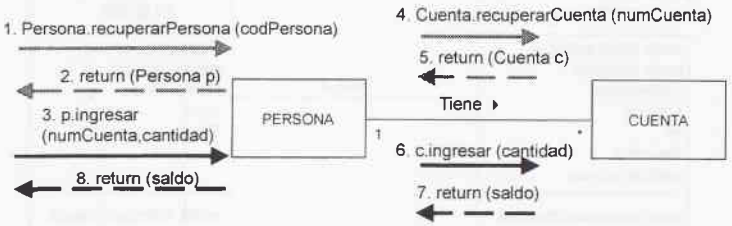



```
class Cuenta {
        //ATRIBUTOS Y CONSTRUCTORES
        // RESTO DE MÉTODOS
        public static boolean eliminarCuenta (int numCuenta) {
                ...
                if (listaCuentas.containsKey(numCuenta)) {
                //Esa cuenta existe
                        listaCuentas.remove(numCuenta);
                        return (true);
                }
                else //Esa cuenta no existe
                        return (false);
}
```



Ejemplo 1

- Ingresar dinero

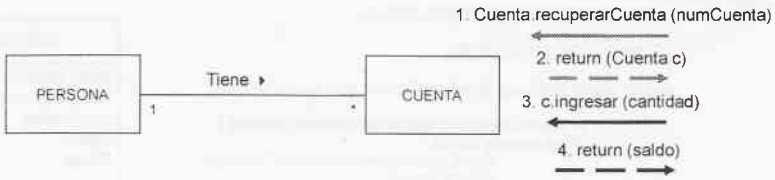


Ejemplo 1

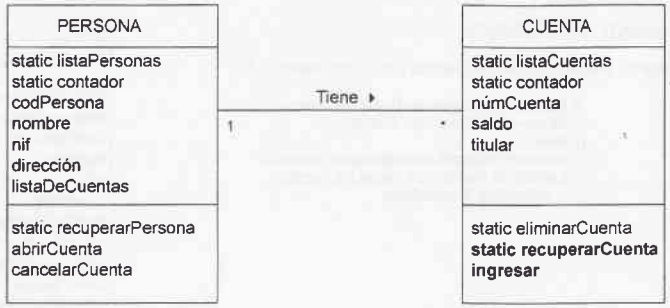
- Ingresar dinero
  - La simulación anterior NO es correcta porque obliga a estar dado de alta en el banco para poder realizar un ingreso.

Ejemplo 1

- Ingresar dinero



Ejemplo 1



Ejemplo 1

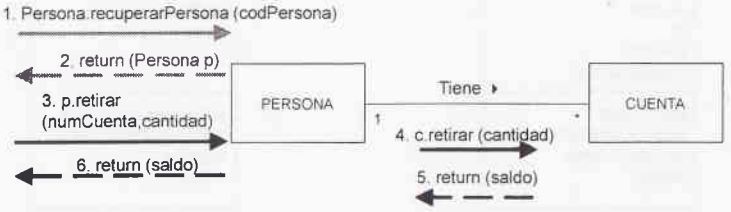
CLASE CUENTA

```
class Cuenta {
        //ATRIBUTOS Y CONSTRUCTORES
        public static Cuenta recuperarCuenta (int numCuenta) {
                ...
                if (listaCuentas.containsKey(numCuenta)) {
                //Esa cuenta existe
                        return (listaCuentas.get(numCuenta));
                }
                else //Esa cuenta no existe
                        return (null);
        }
        public float ingresar (float cantidad ) {
                ...
                saldo = saldo + cantidad;
                return (saldo);
        }
}
```

CUENTA
static listaCuentas static contador númCuenta saldo titular
static eliminarCuenta **static recuperarCuenta** **ingresar**

Ejemplo 1

- Retirar dinero



Ejemplo 1

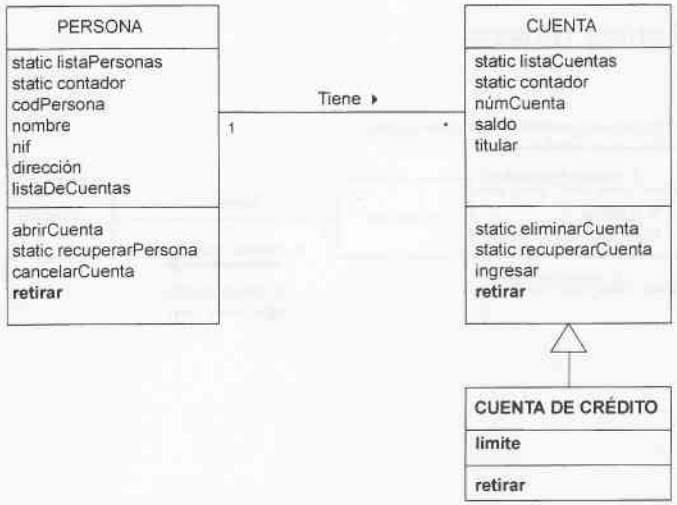
- EL MAIN

```
import java.io.*;

class gestionDeCuentas {

   public static void main (String args[]) {
        boolean salir = false;
        int opcion;
        BufferedReader entrada = new BufferedReader (new InputStreamReader (System.in));
        String nombre;
        String nif;
        String direccion;
        do {
            try {
                System.out.println ("1. Nueva Persona");
                System.out.println ("2. Nueva Cuenta");
                System.out.println ("3. Salir");
                System.out.print ("Opcion? ");

                opcion = (new Integer (entrada.readLine())).intValue();

                switch (opcion) {
                   case 1: System.out.print ("Inserte el nombre ");
                           nombre = entrada.readLine();
                           System.out.print ("Inserte el NIF ");
                           NIF = entrada.readLine();
                           System.out.print ("Inserte la direccion: ");
                           direccion = entrada.readLine();
                           Persona persona = new Persona (nombre, NIF, direccion);
                           break;
                   case 2: //Sentencias para crear una nueva cuenta.
                           break;
                   case 3: salir = true;
                }
            } catch (Exception e) {}//Sentencias que se ejecutan si se presenta la excepcion
        } while (salir != true);
   }
}
```

Ejemplo 1

- Compilación y ejecución
  - Colocamos todas las clases a partir de un mismo directorio.
  - Las compilamos con javac: javac gestionDeCuentas.java
  - Ejecutamos el programa: java gestionDeCuentas

Ejemplo 1

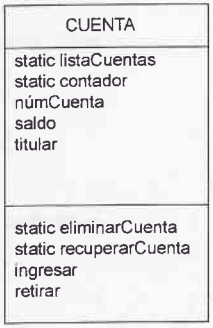


Ejemplo 1

SUPERCLASE CUENTA

En Java tenemos: public, private, protected y package

```
class Cuenta {
        protected static Hashmap<Integer, Cuenta> listaCuentas = new Hashmap<Integer, Cuenta>();
        private static int contador = 0;
        protected int numCuenta;
        protected float saldo;
        protected Persona titular;

        Cuenta (Persona persona) {
                contador++;
                numCuenta = contador;
                saldo = 0;
                titular = persona;
                listaCuentas.put(numCuenta, this);
}
        Cuenta (Persona persona, float saldoInicial) {
                contador++;
                numCuenta = contador;
                saldo = saldoInicial;
                titular = persona;
                listaCuentas.put(numCuenta, this);
        }

        // RESTO DE MÉTODOS
}
```

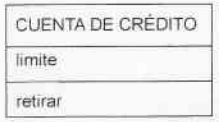


Ejemplo 1

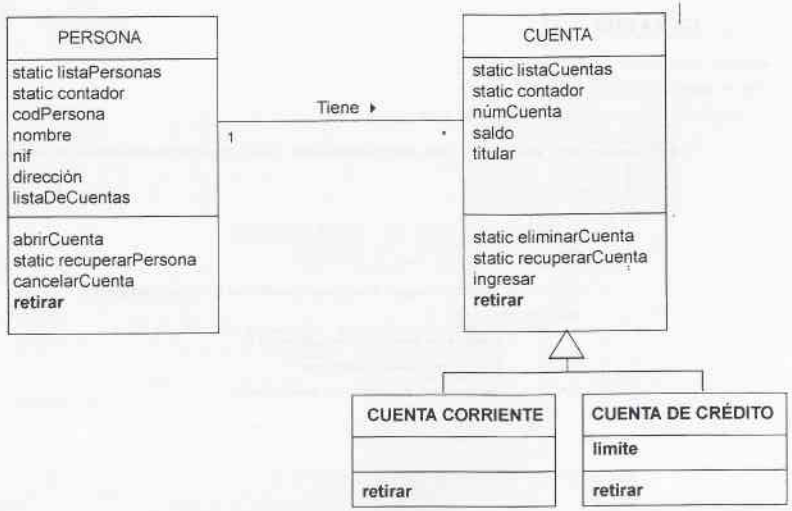
SUBCLASE CUENTA DE CRÉDITO

Super ha de ser la primera instrucción dentro del constructor del hijo.

```
class CuentaCredito extends Cuenta {

        private int limite;

        CuentaCredito (Persona persona, int limiteCuenta) {
                super (persona);  // Llamada al constructor del padre
                limite = limiteCuenta;
        }
        CuentaCredito (Persona persona, float saldoInicial, int limiteCuenta) {
                super (persona, saldoInicial);  // Llamada al constructor del padre
                limite = limiteCuenta;
        }

        public float retirar (float cantidad, int numCuenta) {
                //redefinicion del cuerpo de la funcion
                //teniendo en cuenta el limite
        }
}
```



Ejemplo 1



Ejemplo 1

SUPERCLASE CUENTA

```
class abstract Cuenta {
        // ATRIBUTOS Y CONTRUCTORES
        // RESTO DE METODOS

        public abstract float retirar(float cantidad);
}
```

Tenemos 2 alternativas distintas: o definir el método como abstracto, o definir el método genérico en el padre y redefinirlo en aquellos hijos que cambien.

```
class abstract Cuenta {
        // ATRIBUTOS Y CONTRUCTORES
        // RESTO DE METODOS

        public float retirar (float cantidad){
                //Cuerpo de la funcion
        }
}
```