La herencias es un mecanismo para crear subclases, es decir, refinamientos de clase ya existentes.

1 Cuenta

Las personas pueden abrir cuentas en los bancos.

Las cuentas soportan dos tipos de operaciones: depósitos y giros.

```
class Cuenta {
  int total=0;
  void deposito(int monto) {
    this.total+=monto;
  }
  boolean giro(int monto) {
    this.total-=monto;
    return true; // true=operacion exitosa
  }
  int disponible() {
    return this.total;
  }
}
```

2 CuentaInternacional

Ademas, existe otro tipo de Cuenta, la cuenta internacional, que permite realizar depósitos en dolares, además de las operaciones corrientes.

Java permite crear este nuevo tipo de Cuenta de forma muy elegante:

```
class CuentaInternacional extends Cuenta {
  void depositoEnDolares(int dolares) {
    this.total+=dolares*640;
  }
}
```

Analicemos el código. En la primera línea, se dice que CuentaInternacional es un refinamiento de la clase Cuenta, o lo que es equivalente, que CuentaInternacional extiende a Cuenta. Esto quiere decir que CuentaInternacional “hereda” todos los métodos de Cuenta y los tiene disponibles.

CuentaInternacional tiene, por lo tanto, 4 métodos: depósito, giro, disponible y depositoEnDolares.

Un uso de este mecanismo, que se llama herencia, es crear nuevas clases que agregan nuevos métodos a clases ya existentes, como se vio en este ejemplo.

3 CuentaSegura

Una cuenta segura es aquella que
pesos.

En este caso, podemos extende
agregar un método, vamos a modi

```
class CuentaSegura exte
  boolean giro(int mont
    if (this.total-mont
       return false;
    this.total-=monto;
    return true;
  }
}
```

En este caso, lo que se le indica a
la clase Cuenta, salvo por el méto
de 1 millón de pesos”.

Si observas con atención, te da
lar al método giro original. Sería
afortunadamente Java lo permite

```
class CuentaSegura exte
  boolean giro(int mont
    if (this.total-mont
       return false;
    return super.giro(m
  }
}
```

super.giro es el método giro que f
“renombrado” internamente para

4 CuentaSeguraPref

Existe un cuarto tipo de Cuenta,
monto en el que uno se puede end

```
class CuentaSeguraPrefe
  int maximaDeuda;
  void setMaximaDeuda(i
```

```
  boolean giro(int monto) {
    if (this.total-monto<-this.maximaDeuda)
      return false;
    return super.giro(monto);
  }
}
```

A diferencia de los ejemplos anteriores, en esta clase

- se redefine un método (giro)
- se agrega un método (setMaximaDeuda)
- se agrega una variable de instancia (maximaDeuda)

5 Tipo estático y tipo dinámico

Cada variable que hace referencia a un objeto en Java tiene 2 tipos:

estático el tipo declarado para la variable

dinámico el tipo del objeto referenciado por la variable

Por ejemplo, en el caso de

```
Cuenta c;
c=new Cuenta();
```

el tipo estático de c es Cuenta (primera línea), y el tipo dinámico también lo es (c, en la segunda línea, hace referencia a un objeto Cuenta).

Parece bastante trivial, pero no siempre es así:

```
Cuenta c=new CuentaSegura();
```

En este caso, el tipo estático de c es Cuenta y su tipo dinámico es CuentaSegura. No son iguales!

Todo el poder real de la orientación a objetos nace de las siguientes reglas:

1. El tipo dinámico es igual al tipo estático o subclase de este
2. Los métodos invocados sobre un objeto tienen que estar declarados en el tipo estático
3. El método invocado es el declarado en el tipo dinámico

Veamos un par de ejemplos de estas 3 leyes:

```
c.deposito(100);
boolean b=c.giro(200000
```

La primera línea cumple la 1ra ley.
bigüedad. En la tercera línea, el m
invocado es el definido en la clas
permite terminar con una deuda ta

5.2 Cuenta y CuentaInte

```
Cuenta c=new CuentaInte
c.depositoEnDolares(1);
```

Este ejemplo, en cambio, ni siquie
encuentra definido en el tipo estát

6 Aplicación

Considera que el Banco Beauche
sus clientes. Algunas son Cuentas
gurasPreferenciales. La gran may

Se te pide escribir un pedazo d
los clientes, respetando sus tipos d

La solución no deja de ser elega

```
Cuenta c[]=new Cuenta[1
c[0]=new Cuenta();
c[1]=new CuentaInternac
...
// el pedazo de código
for (int i=0;i<c.length
  c[i].giro(100);
```