STATE
pattern

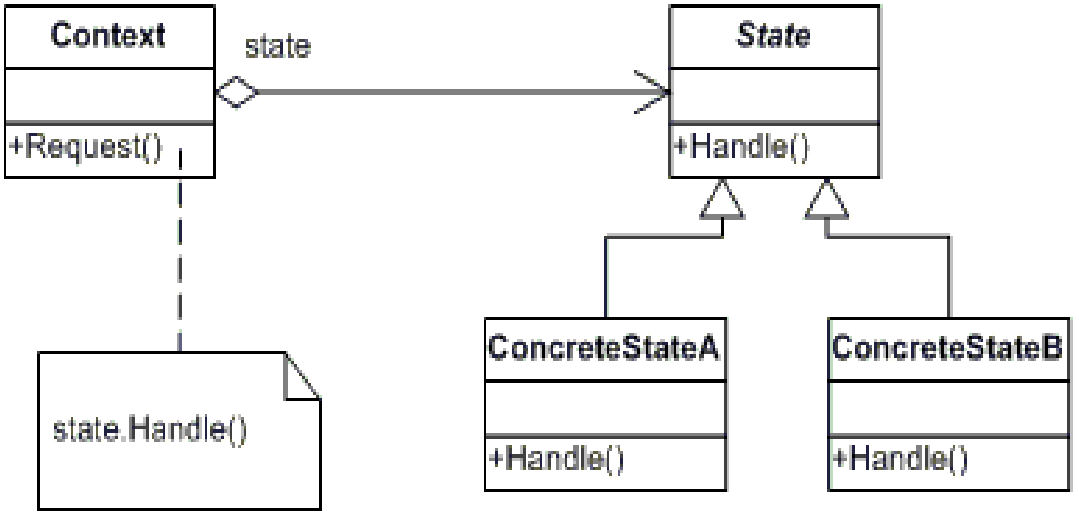
Artur Juvé Vidal
Enginyeria del Software I
EINF 2006-2007

Definicions (1/3)

- El patró State s'utilitza quan un objecte pot adquirir diferents estats interns i el seu comportament pot variar depenent de l'estat en què es troba.
- L'estat d'un objecte són els seus atributs i el comportament d'un objecte són els seus mètodes.
- El patró State proposa delegar el comportament a objectes de tipus State, objectes que alhora defineixen l'estat.

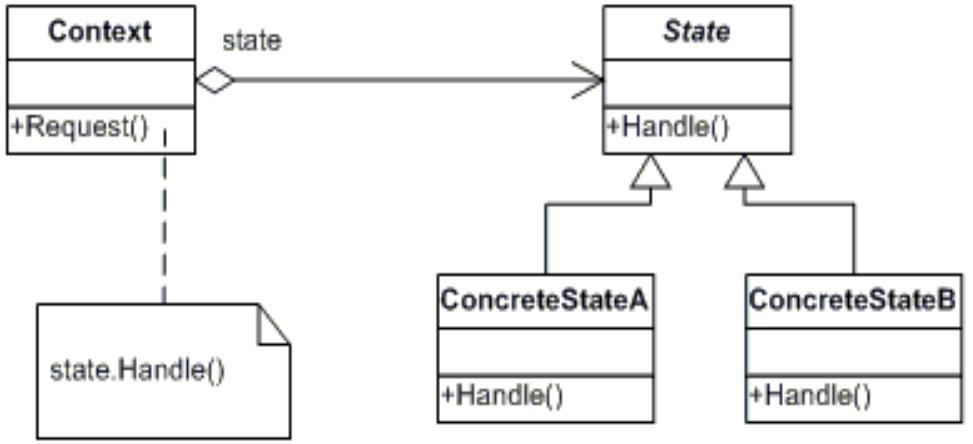
Definicions (2/3)

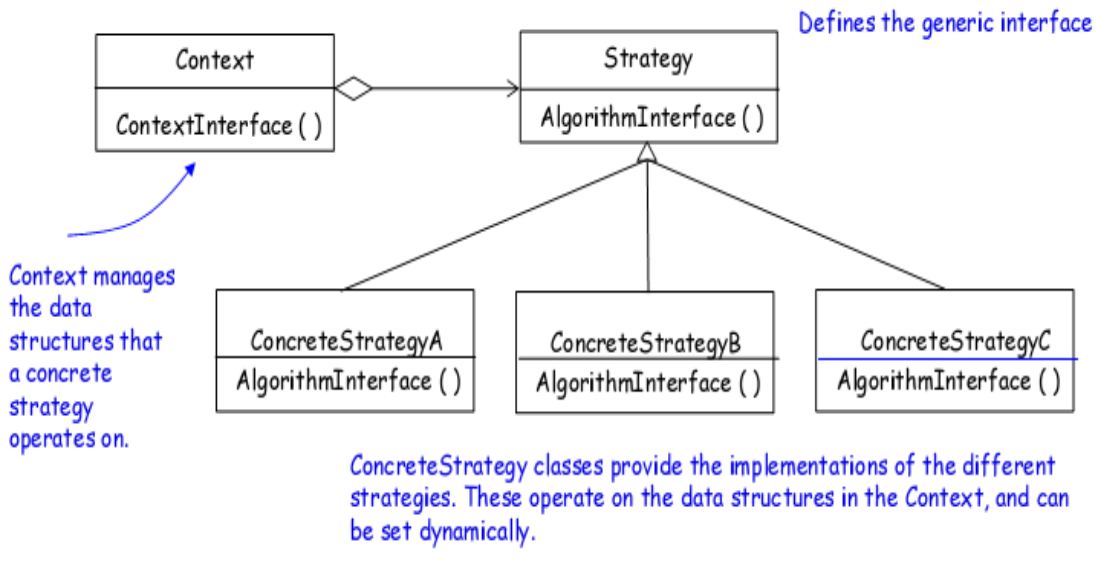
- Un objecte de la classe Context pot modificar el seu comportament quan aquest canvia d'estat.
- El comportament de la classe Context s'encapsula en una família d'estats.



Definicions (3/3)

- State vs Strategy
  - L'esquema de classes del patró State és idèntic del patró Strategy. El patró State difereix de Strategy en ús i finalitat.





Exemple1 (1/3)

- Exemple d'un editor de HTML.
- L'editor permet visualitzar la pàgina en diferents modes:
  - Design view, HTML view i Quick page view.
- Si un usuari canvia el mode de visualitzar la pàgina HTML llavors l'editor ha de canviar el seu comportament.

Exemple1 (2/3)

- Una primera solució seria controlar el comportament amb una estructura condicional.
- Aquest disseny no compleix el principi OCP.

```
public Context{
            …
            …
  someMethod(){
    if (state_1){
      //do something
    }else if (state_2){
      //do something else
    }
            …
            …
  }
            …
            …
}
```

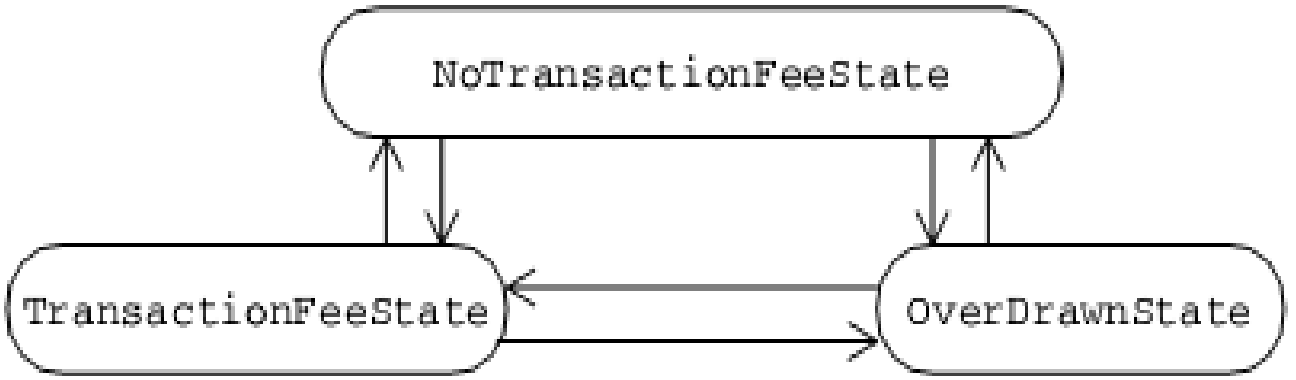
Exemple1 (3/3)

- La solució utilitzant el patró State evita una estructura condicional en el codi.
- Aquest disseny compleix el principi OCP.

```
public Context{
            …
            …
  someMethod(){
    objCurrentState.someMethod();
  }
            …
            …
}
```

Exemple2 (1/9)

- Exemple del compte corrent bancari d'un negoci amb saldo descobert.
- Es suposen tres estats possibles del compte:
  - No transaction fee state
  - Transaction fee state
  - Overdrawn state

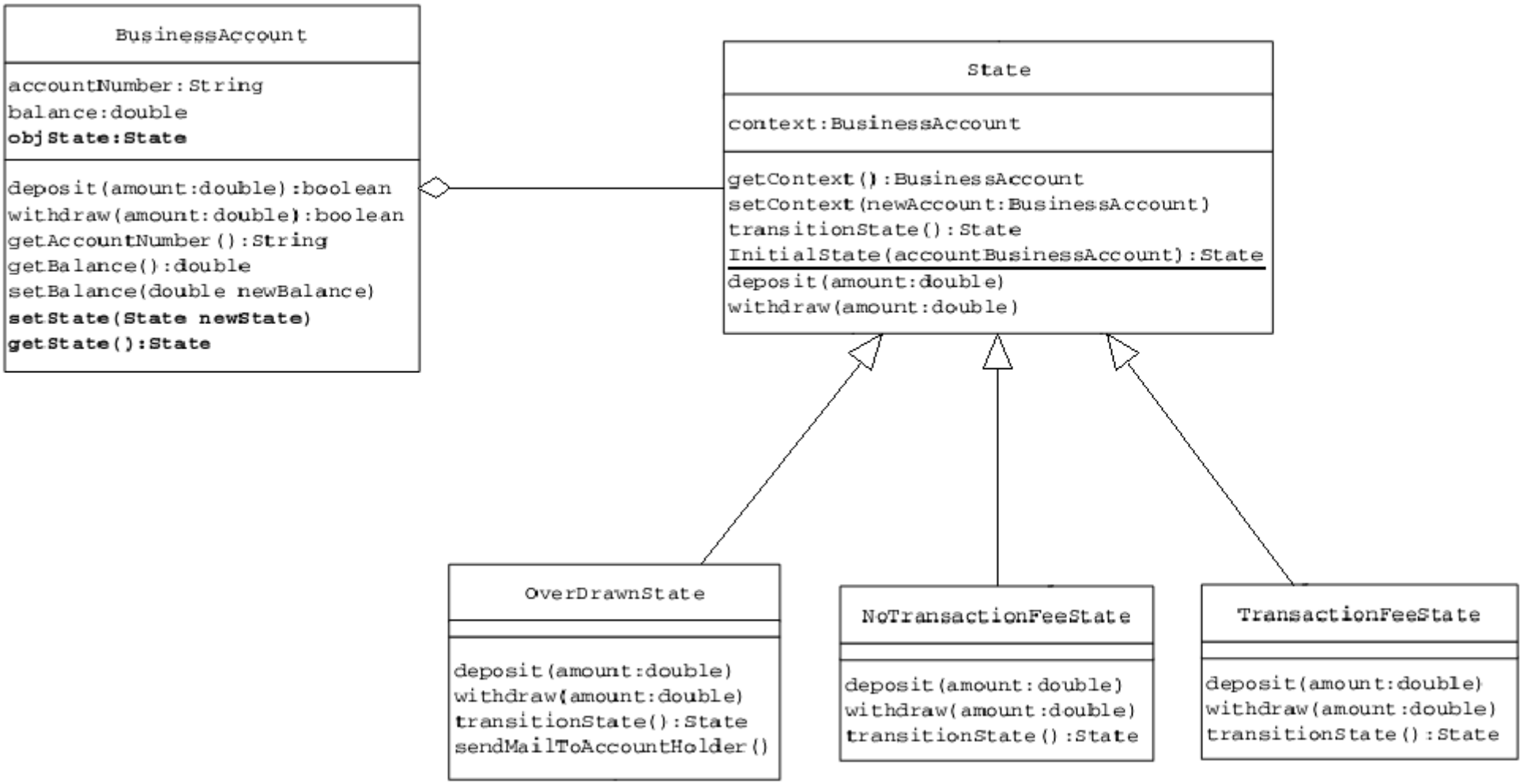


Exemple2 (2/9)

- Els canvis d'estat són:

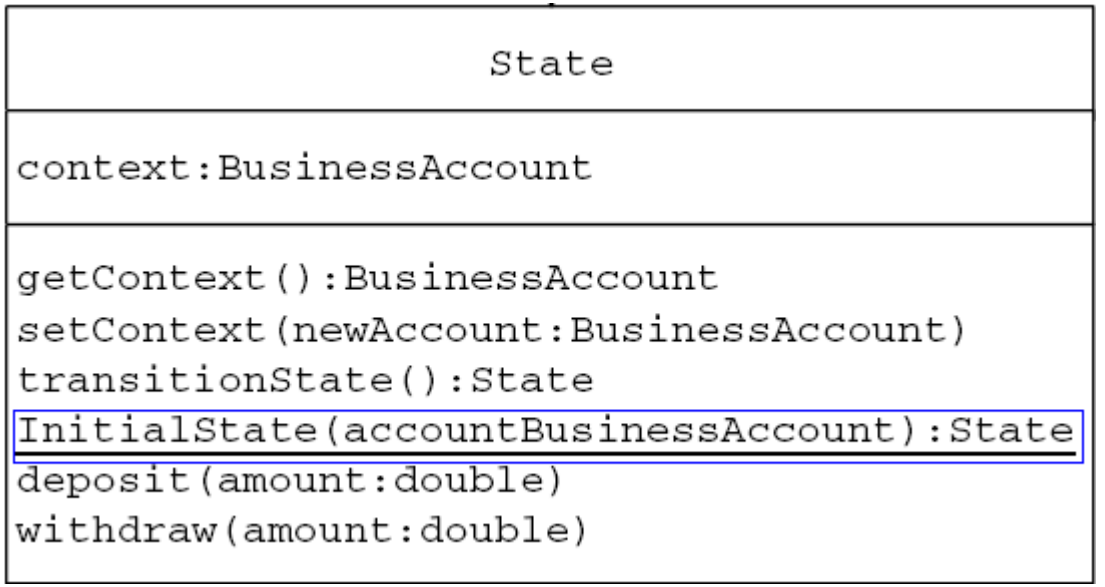
From	*To*	*What Causes the Transition*
No transaction fee state	Transaction fee state	A withdrawal that can make the balance positive but less than the minimum balance.
	Overdrawn state	A withdrawal that can make the balance negative.
Transaction fee state	No transaction fee state	A deposit that can make the balance greater than the minimum balance.
	Overdrawn state	A withdrawal that can make the balance negative.
Overdrawn state	No transaction fee state	A deposit that can make the balance greater than the minimum balance.
	Transaction fee state	A deposit that can make the balance positive but less than the minimum balance.

Exemple2 (3/9)


BusinessAccount
accountNumber:String
balance:double
objState:State
deposit(amount:double):boolean
withdraw(amount:double):boolean
getAccountNumber():String
getBalance():double
setBalance(double newBalance)
setState(State newState)
getState():State
State
context:BusinessAccount
getContext():BusinessAccount
setContext(newAccount:BusinessAccount)
transitionState():State
InitialState(accountBusinessAccount):State
deposit(amount:double)
withdraw(amount:double)
OverDrawnState
deposit(amount:double)
withdraw(amount:double)
transitionState():State
sendMailToAccountHolder()
NoTransactionFeeState
deposit(amount:double)
withdraw(amount:double)
transitionState():State
TransactionFeeState
deposit(amount:double)
withdraw(amount:double)
transitionState():State

Exemple2 (4/9)

- La classe State conté un mètode estàtic que retorna l'estat inicial.




```
public static State InitialState(BusinessAccount account) {
  return new NoTransactionFeeState(account);
}
```

Exemple2 (5/9)

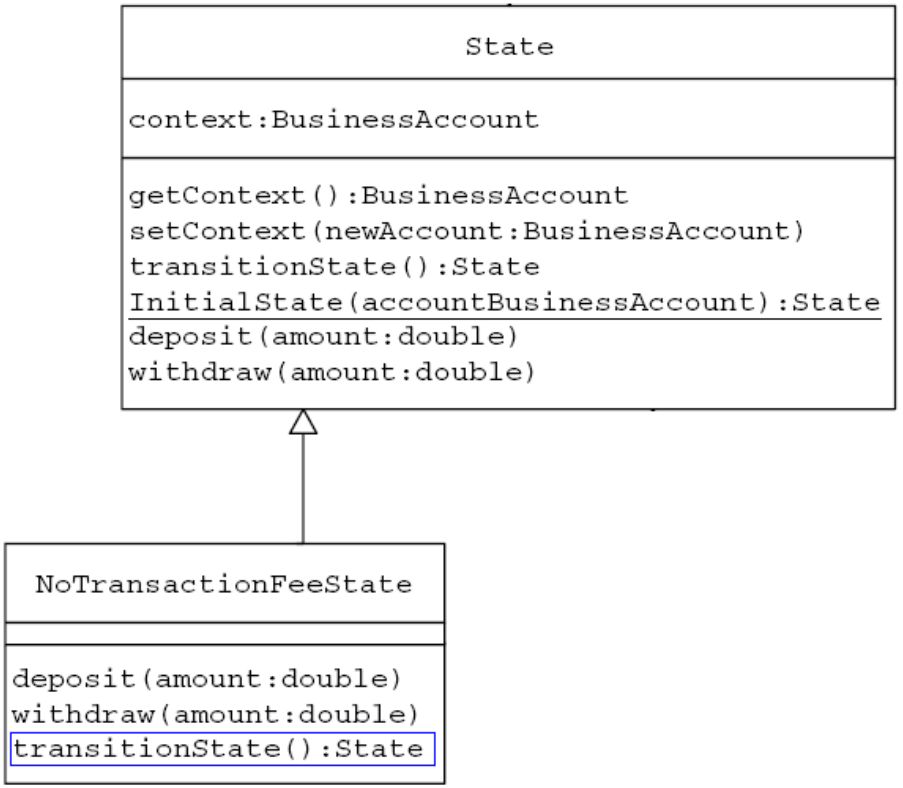
- En crear una instància de BusinessAccount s'inicialitza a l'estat inicial.

```
public BusinessAccount(String accountNum) {
  accountNumber = accountNum;
  objState = State.InitialState(this);
}
```

- La transició d'estats és independent de la instància BusinessAccount llavors State ha de conèixer BusinessAccount per accedir a la informació necessària pel canvi d'estat.

Exemple2 (6/9)

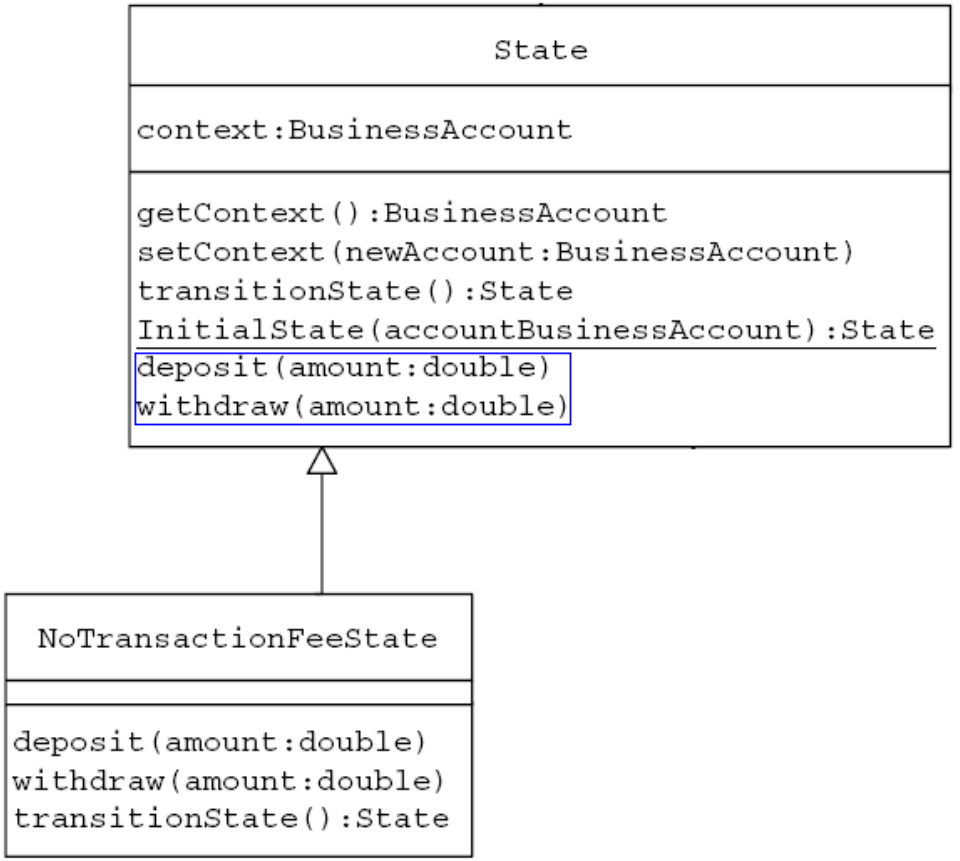
- El mètode de transició és particular de cada tipus d'estat.




```
public State transitionState() {
  double balance = getContext().getBalance();
  if (balance < 0) {
    getContext().setState(new OverDrawnState(this));
  } else {
    if (balance >= BusinessAccount.MIN_BALANCE) {
      getContext().setState(
        new NoTransactionFeeState(this));
    }
  }
  return getContext().getState();
}
```

Exemple2 (7/9)

- El mètode de transició és cridat en mètodes que poden generar canvis en l'estat (deposit(...) i withdraw(...)).




```
public boolean deposit(double amount) {
  double balance = getContext().getBalance();
  getContext().setBalance(balance + amount);
  transitionState();
  System.out.println("An amount " + amount +
                     " is deposited ");
  return true;
}
public boolean withdraw(double amount) {
  double balance = getContext().getBalance();
  getContext().setBalance(balance - amount);
  transitionState();
  System.out.println("An amount " + amount +
                     " is withdrawn ");
   return true;
}
```

Exemple2 (8/9)

- La família State utilitza en excés el polimorfisme de mètodes fent més difícil entendre el codi. Es fa un ús excessiu de l'herència.

State Class

```
public boolean withdraw(double amount) {
  double balance = getContext().getBalance();
  getContext().setBalance(balance - amount);
  transitionState();
  System.out.println("An amount " + amount +
                     " is withdrawn ");
   return true;
}
```

NoTransactionFeeState Class

```
public boolean withdraw(double amount) {
  double balance = getContext().getBalance();
  if ((balance - amount) >
      BusinessAccount.OVERDRAW_LIMIT) {
    super.withdraw(amount);
    return true;
  } else {
    System.out.println(
      BusinessAccount.ERR_OVERDRAW_LIMIT_EXCEED);
    return false;
  }
}
```

Exemple2 (9/9)

- No respecta el principi OCP. Si s'afegeixen nous estats segurament s'haurà de modificar el codi dels mètodes de transició.
- A favor, aquest sistema es controla sol i no és necessari controlar externament l'estat.
- Per tant, el sistema automàtic d'estats resulta interessant si es preveu no s'hauran d'afegir nous estats.