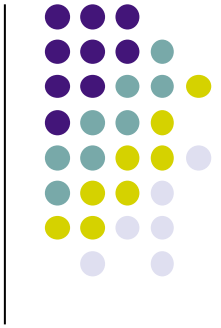
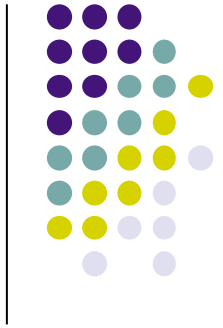
Example:
class Account



```
class Account
   attr balance:0
   meth transfer(Amount)
        balance := @balance+Amount
   end
   meth getBal(B)
        B=@balance
   end
end
A={New Account transfer(100)}
```

Conservative extension

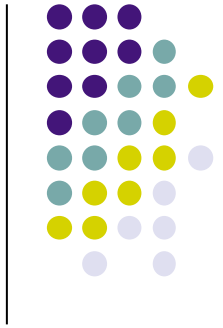


VerboseAccount:
An account that displays all transactions

```
class VerboseAccount
    from Account
    meth verboseTransfer(Amount)
            …
    end
end
```

The class VerboseAccount has methods transfer, getBal and the new method verboseTransfer.

Nonconservative extension

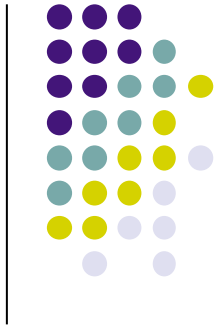


AccountWithFee:
An account with a fee

```
class AccountWithFee
    from VerboseAccount
    attr fee:5
    meth transfer(Amount)
            …
    end
end
```

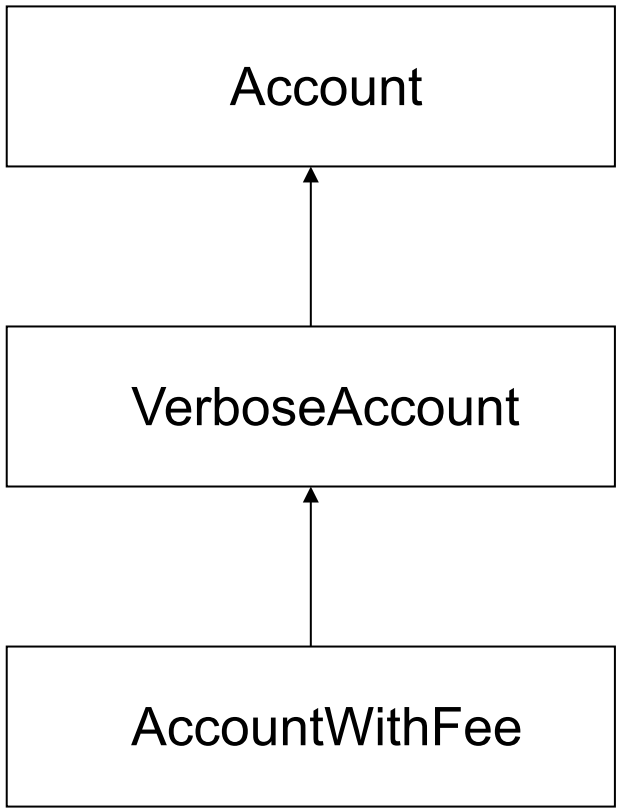
The class AccountWithFee has methods transfer, getBal and verboseTransfer.
The transfer method has been overridden.

Class hierarchy



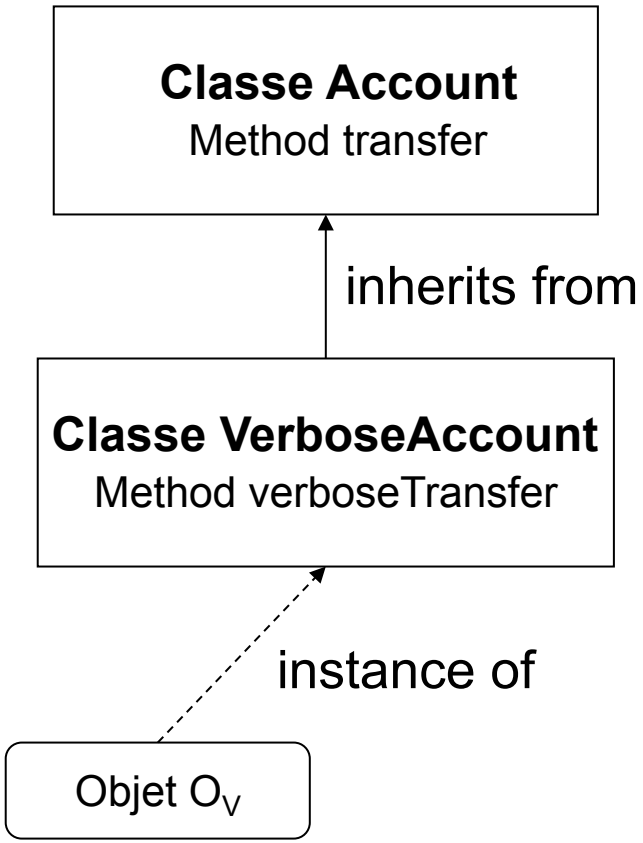
```
class VerboseAccount
    from Account
    meth verboseTransfer(Amount)
            …
    end
end

class AccountWithFee
    from VerboseAccount
    attr fee:5
    meth transfer(Amount)
            …
    end
end
```

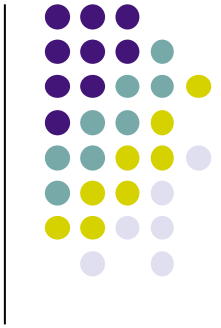


Dynamic link

- Let us define the new method verboseTransfer
- In the definition of verboseTransfer, we need to call transfer
- Syntax: {**self** transfer(A)}
  - The transfer method is chosen in the class of the calling object O_V
  - **self** = the calling object, instance of VerboseAccount



Definition of VerboseAccount

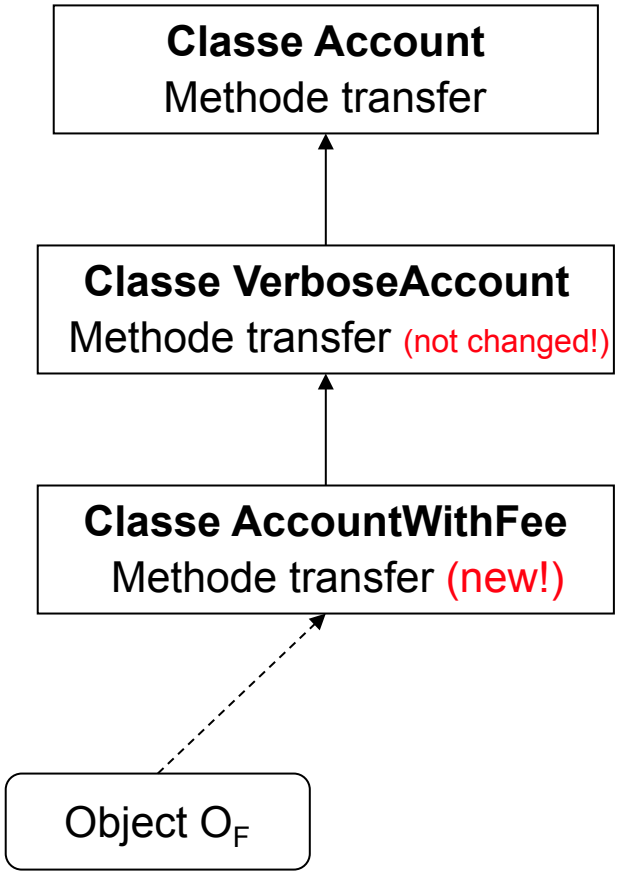


```
class VerboseAccount
    from Account
    meth verboseTransfer(Amount)
            {self transfer(Amount)}
            {Browse @balance}
    end
end
```

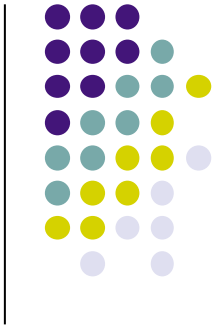
The class VerboseAccount has methods transfer, getBal and verboseTransfer.

Static link

- Let us override the old transfer method in AccountWithFee
- In the new transfer method, we need to call the old method!
- Syntax: VerboseAccount,transfer(A)
  - The class containing the old definition has to be named!
  - The transfer method is taken from the class VerboseAccount



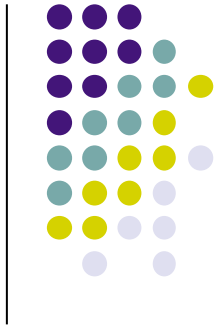
Definition of AccountWithFee



```
class AccountWithFee
    from VerboseAccount
    attr fee:5
    meth transfer(Amt)
            VerboseAccount,transfer(Amt-@fee)
    end
end
```

The class AccountWithFee has methods transfer, getBal and verboseTransfer. The transfer method has been overridden.

The magic of dynamic links

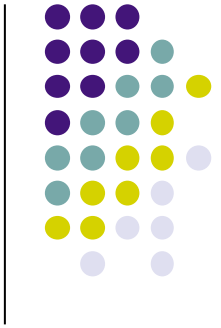


- Look at the following fragment:

  A={New AccountWithFee transfer(100)}
  {A verboseTransfer(200)}

- What does it do?
  - Which transfer method is called by verboseTransfer?
    - The old one or the new one?
  - Observe: when VerboseAccount was defined, the class AccountWithFee did not exist yet
- Answer: !!

Example of a dynamic link



```
meth verboseTransfer(Amount)
    {self transfer(Amount)}
    {Browse @balance}
end
```

Call 1:

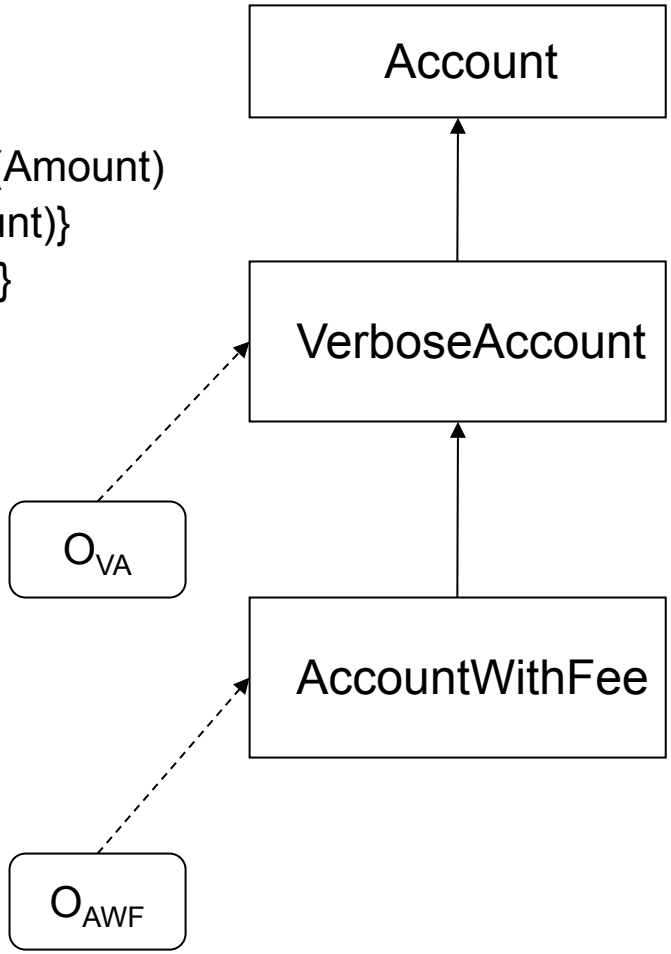
$\{O_{VA}$ verboseTransfer(200)$\}$

Which transfer method?

Call 2:

$\{O_{AWF}$ verboseTransfer(200)$\}$

Which transfer method?

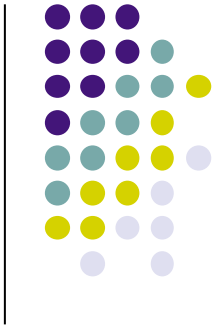


Account
getBal(B)
transfer(Amt)

VerboseAccount
getBal(B)
transfer(Amt) % old definition
verboseTransfer(Amt)

AccountWithFee
getBal(B)
transfer(Amt) % new definition
verboseTransfer(Amt)

Nonconservative extension



Danger!
The invariants
are broken.

```
class AccountWithFee
    from VerboseAccount
    attr fee:5
    meth transfer(Amt)
            VerboseAccount,transfer(Amt-@fee)
    end
end
```

Invariant:

{A getBal(B)}

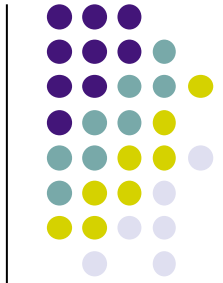
{A transfer(S)}

{A getBal(B1)}

% Is B1=B+S ?

% **No! It's broken!**

Summary of static and dynamic links



- The goal of static and dynamic links is to choose which method to execute
- Dynamic link: {**self** M}
  - The method is chosen in the class of the object
  - This class is only known during execution, this is why it is called a *dynamic* link
  - It should *always* be used by default
- Static link: SuperClass,M
  - The method is chosen in SuperClass
  - This class is known during compilation (it is SuperClass), this is why it is called a *static* link
  - It is *only* needed for overriding an existing method
  - When a method is overridden, the new definition often has to access the old one, and it uses a static link to do this