Structural design pattern

Vũ Hoàng Phi
Software Development 3

VC Development Center Vietnam

Hanoi, 10/2018

Be First, Do It Right, Work Smart

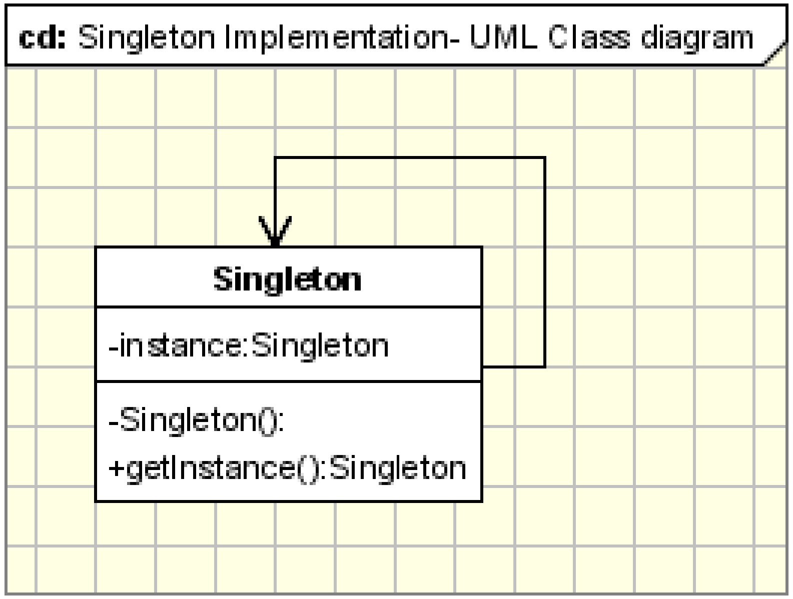
LG Life's Good

Agenda

- Design pattern overview
- Adapter
- Façade
- Proxy
- Conclusion
- Discussion

Design pattern overview

- Design patterns are optimized, general solutions
- Used to solve programming problem which met daily
- DPs are not for specific language. Just often in OOP
- DPs in OOP show relations and communications between objects/classes
- Maybe you have applied DPs in your project but don’t know their name



Design pattern overview

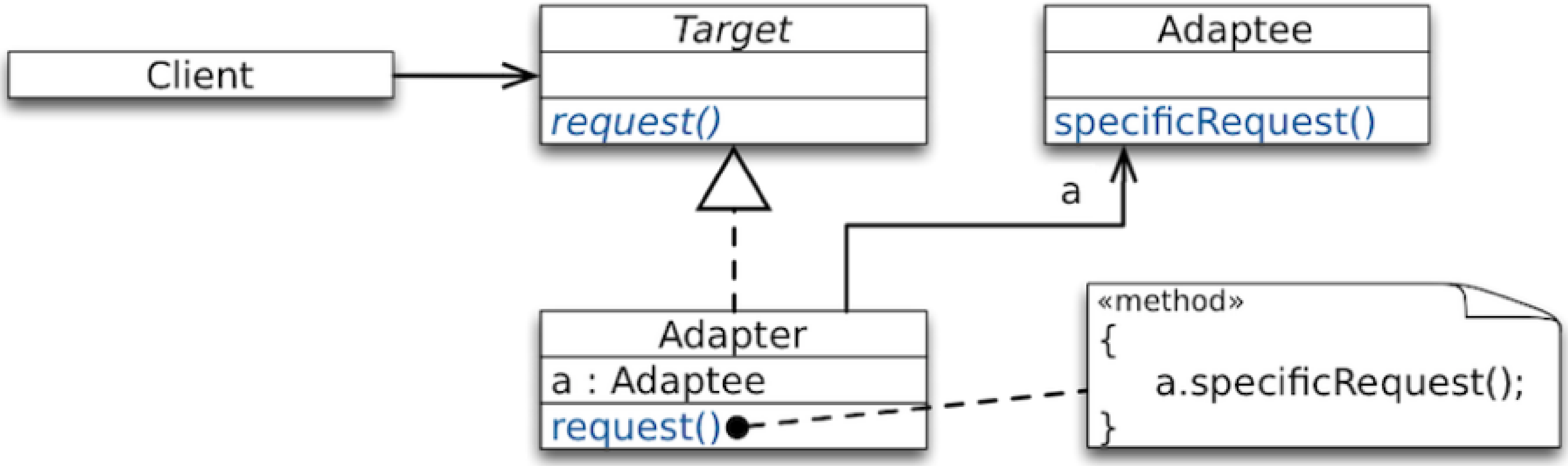
- Advance:
  - Reusability
  - Expandability
  - Communicability
- Structural pattern
  - Defining relationship between objects/classes
  - Called Wrapper classes informally

Adapter - Intent

- Fit foreign components into an existing design
- Adapter pattern makes classes work together although they have incompatible interface

Adapter – class diagram

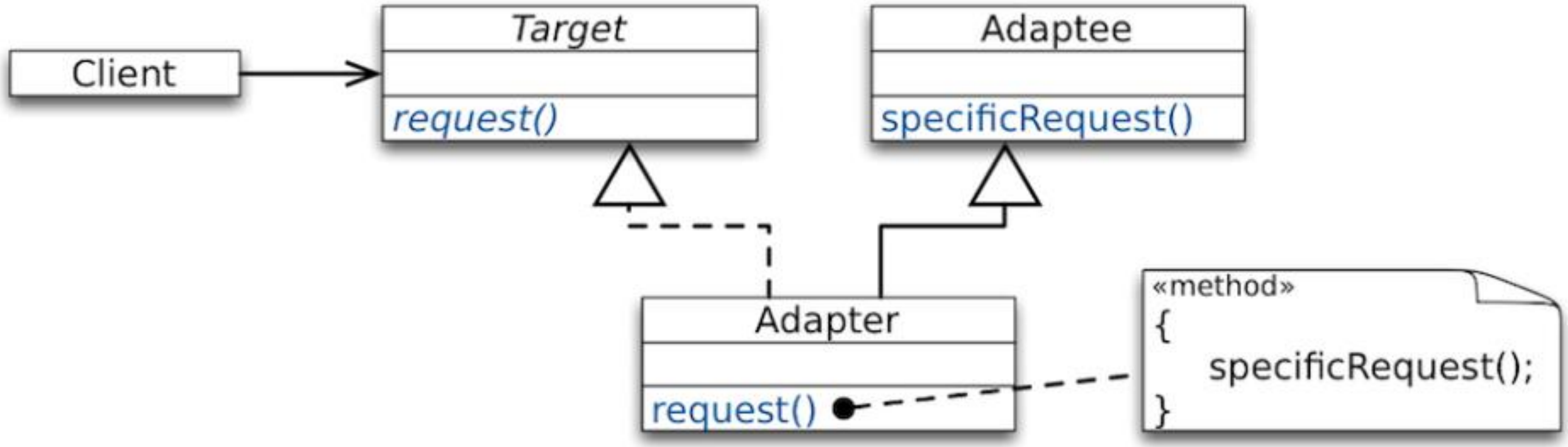
- Object adapter



LG Life's Good

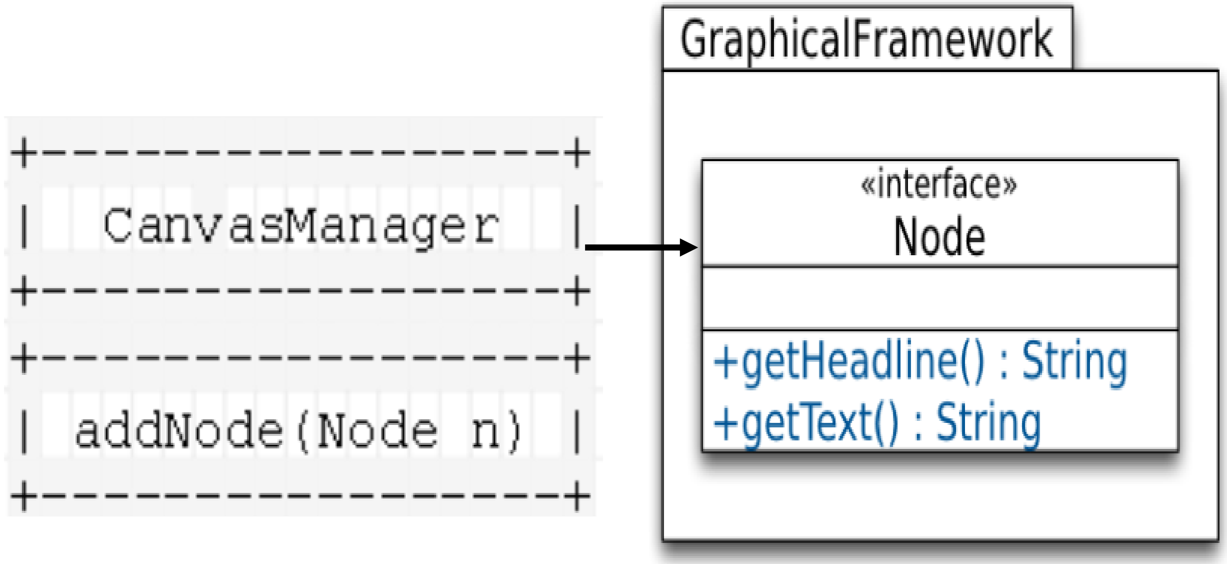
Adapter – class diagram

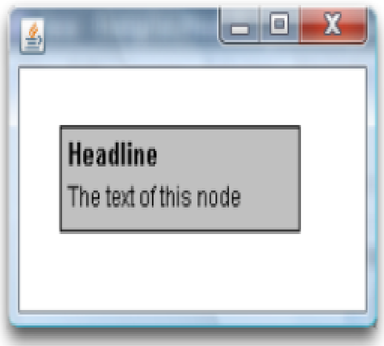
- Class adapter



Adapter – case study

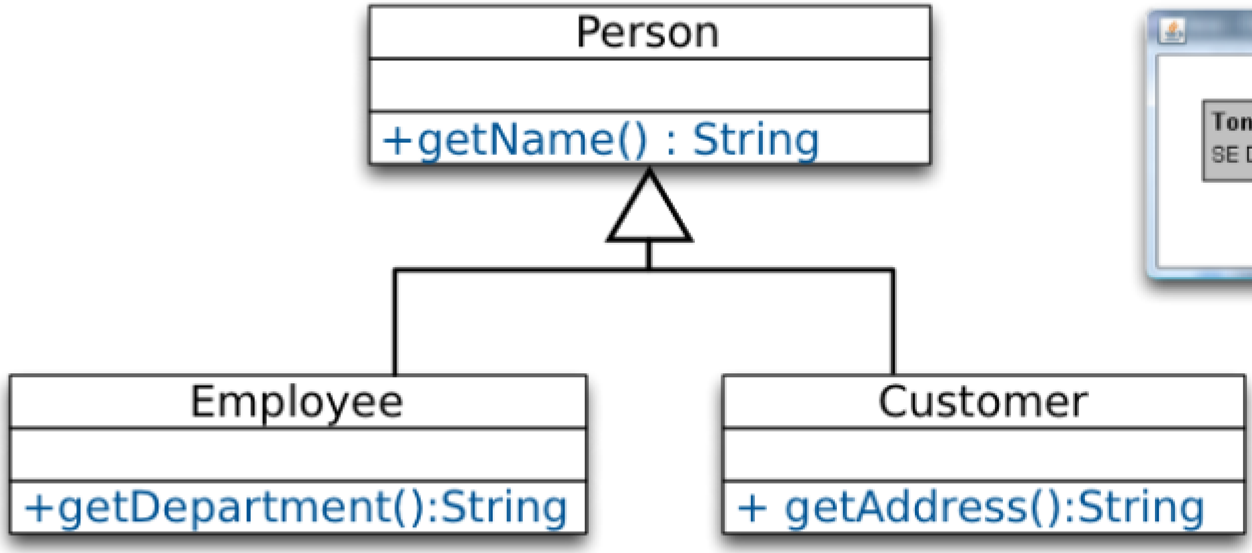
- GraphicFramework provides an interface named “Node” defining a Card item behaviour

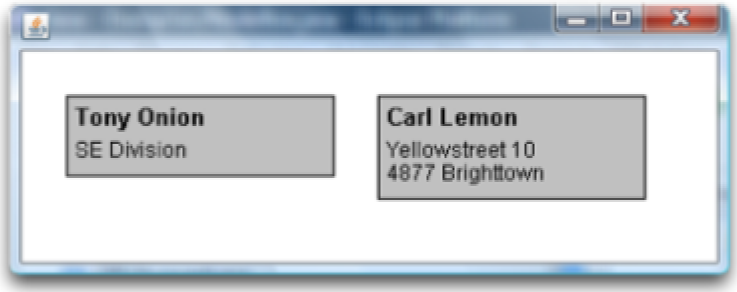




Adapter – case study

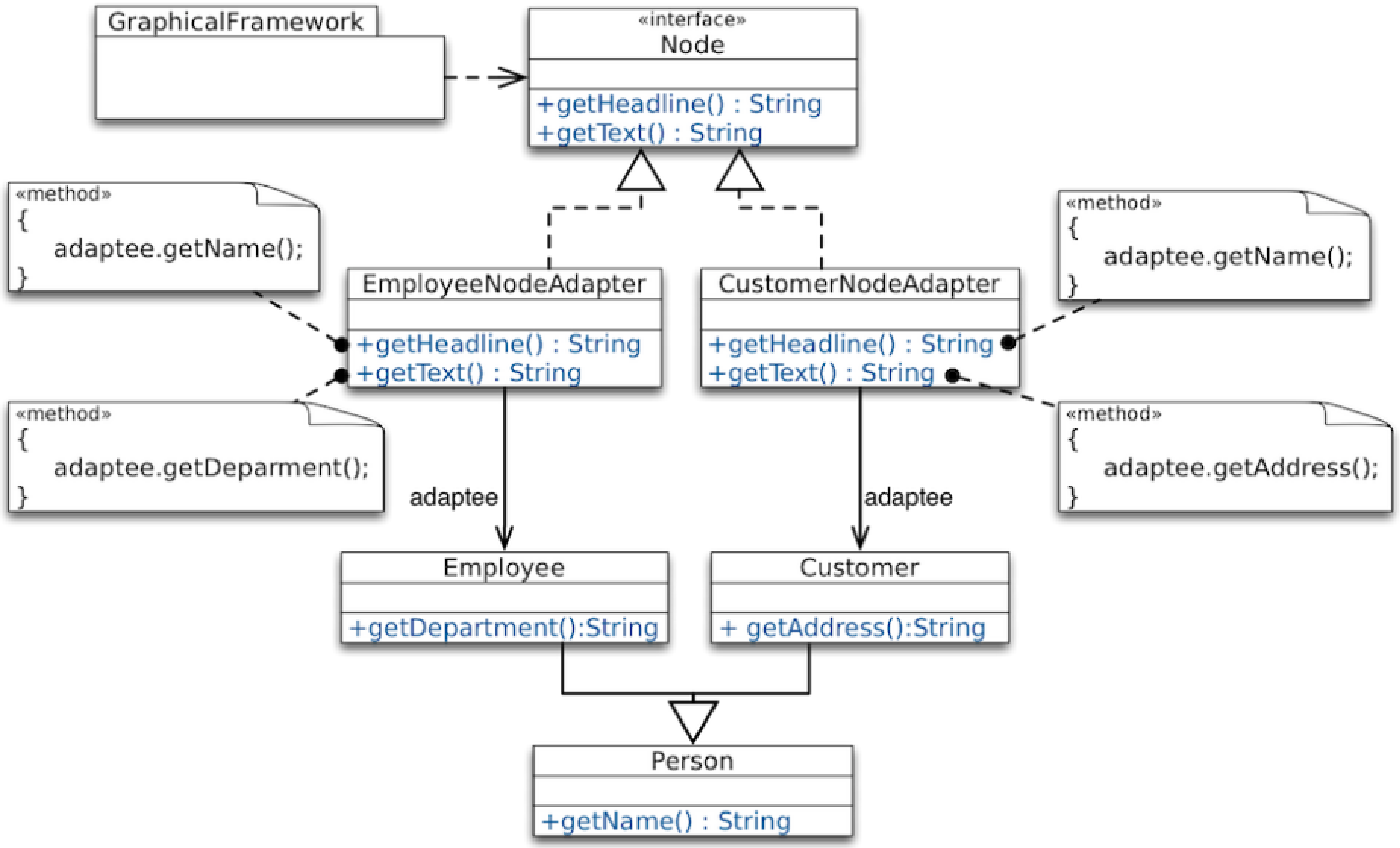
- Application feature allows showing both Employee and Customer information as Card items





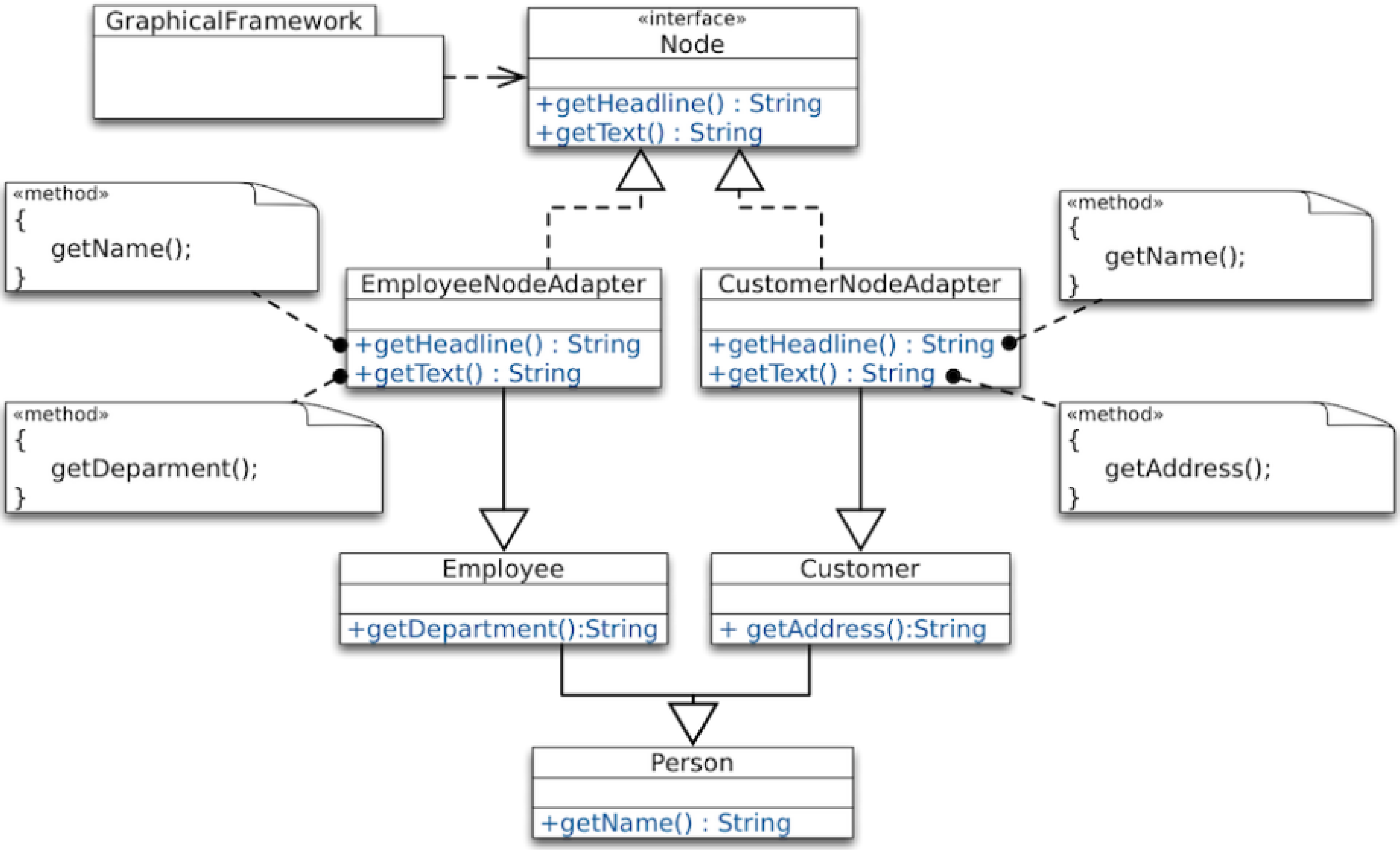
Adapter – case study

- Using object adapter



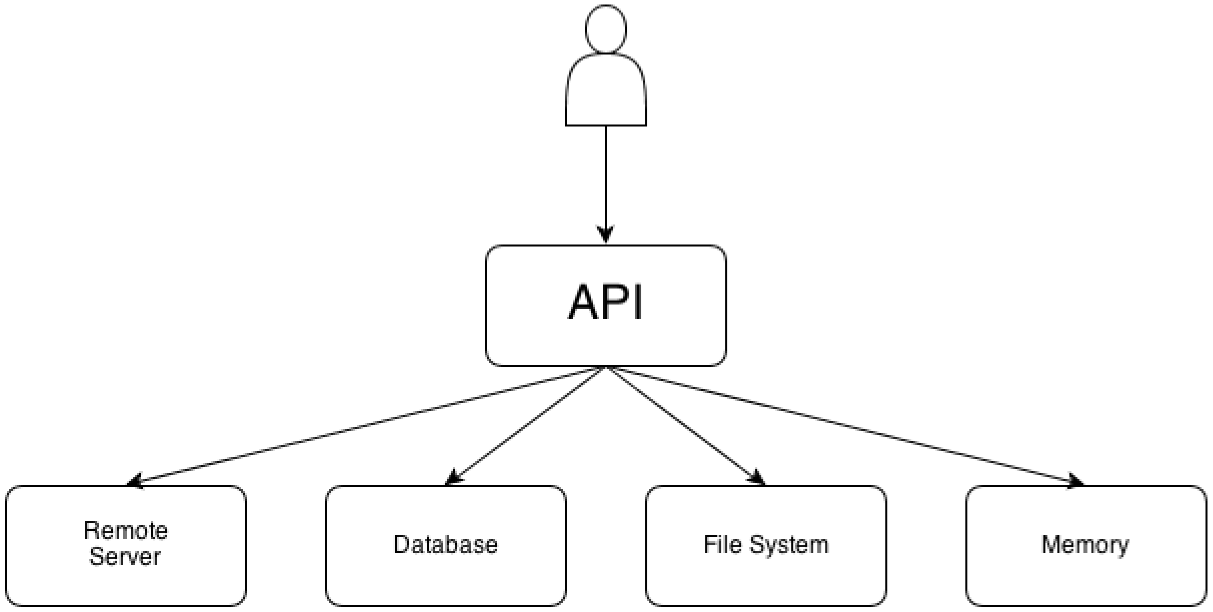
Adapter – case study

- Using class adapter

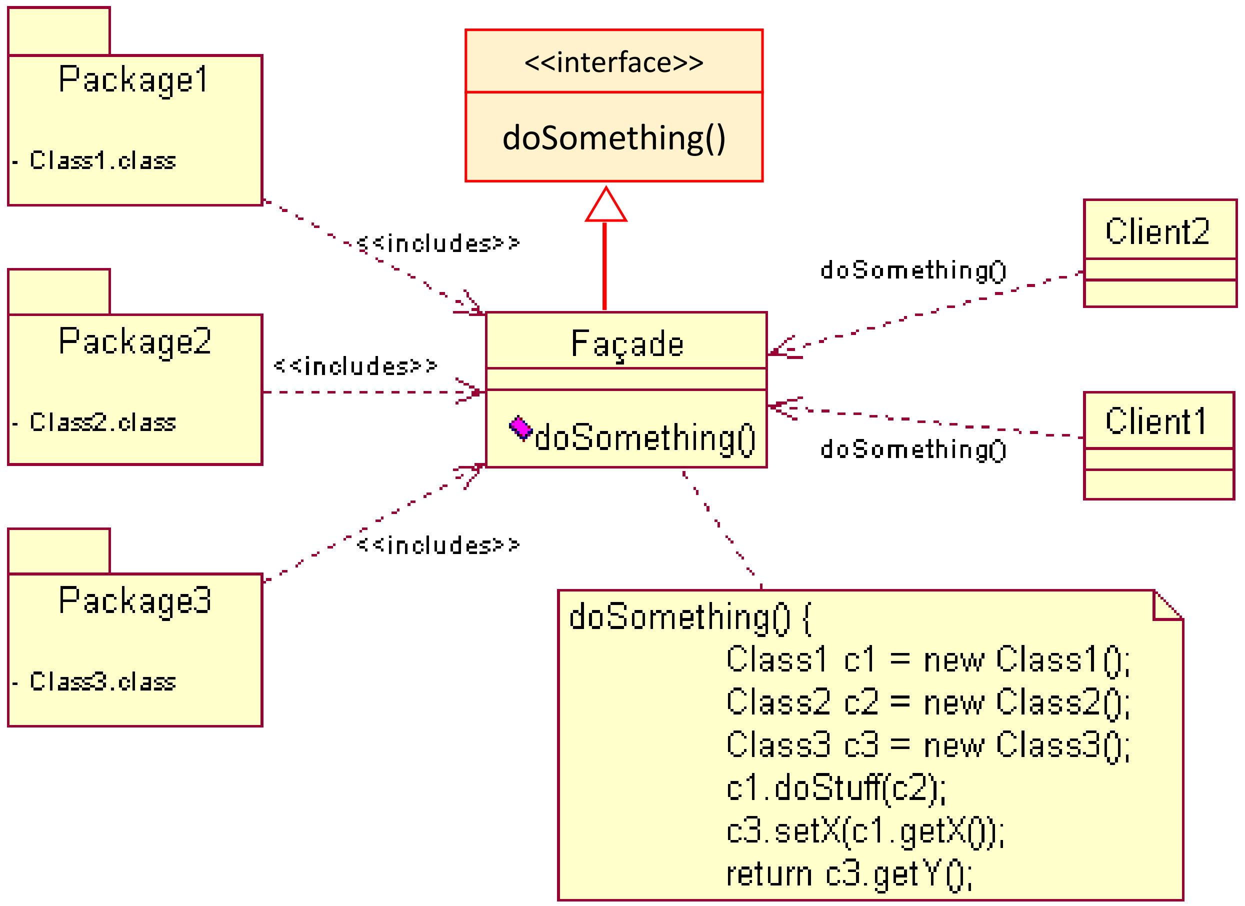


Façade - Intent

- Provides high level interface easy-to-use
- Hides the complexity of system to client



Façade – class diagram

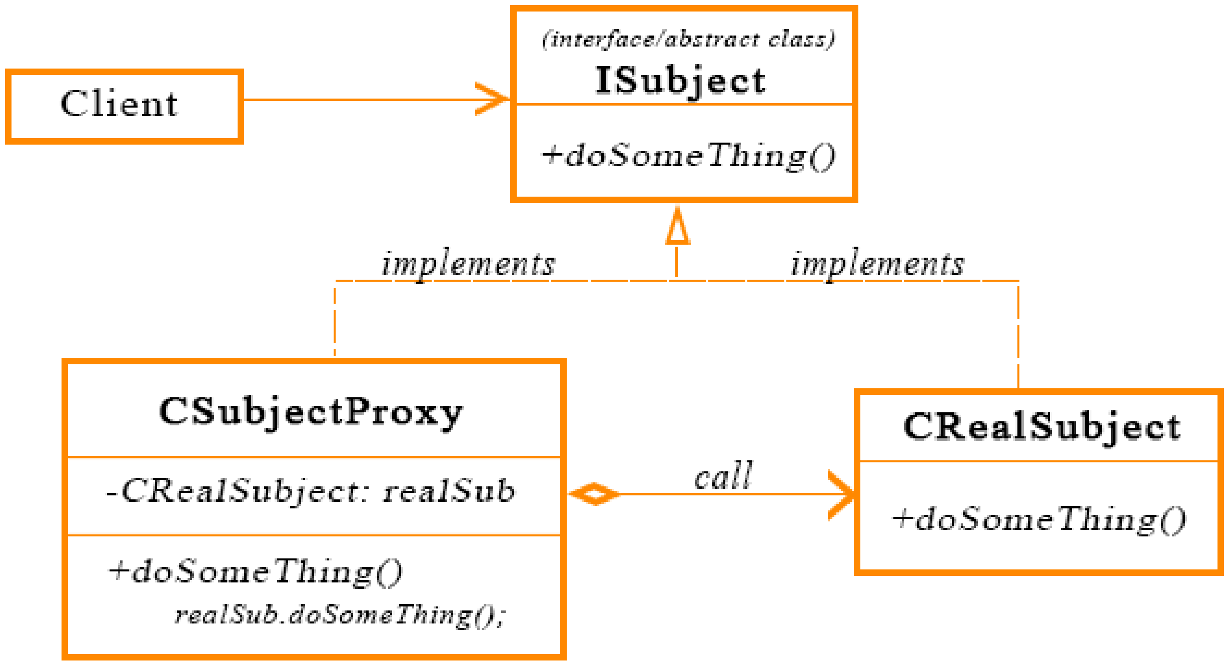


```
doSomething() {
        Class1 c1 = new Class1();
        Class2 c2 = new Class2();
        Class3 c3 = new Class3();
        c1.doStuff(c2);
        c3.setX(c1.getX());
        return c3.getY();
```

Proxy - Intent

- Provides a representative for an object
- Virtual proxy:
  - Used when accessing objects having complex structure or contain huge data (image, voice, dataset …)
- Protection proxy
  - Applying some filter on access right for real data
- Remote proxy
  - Accesing remoted object

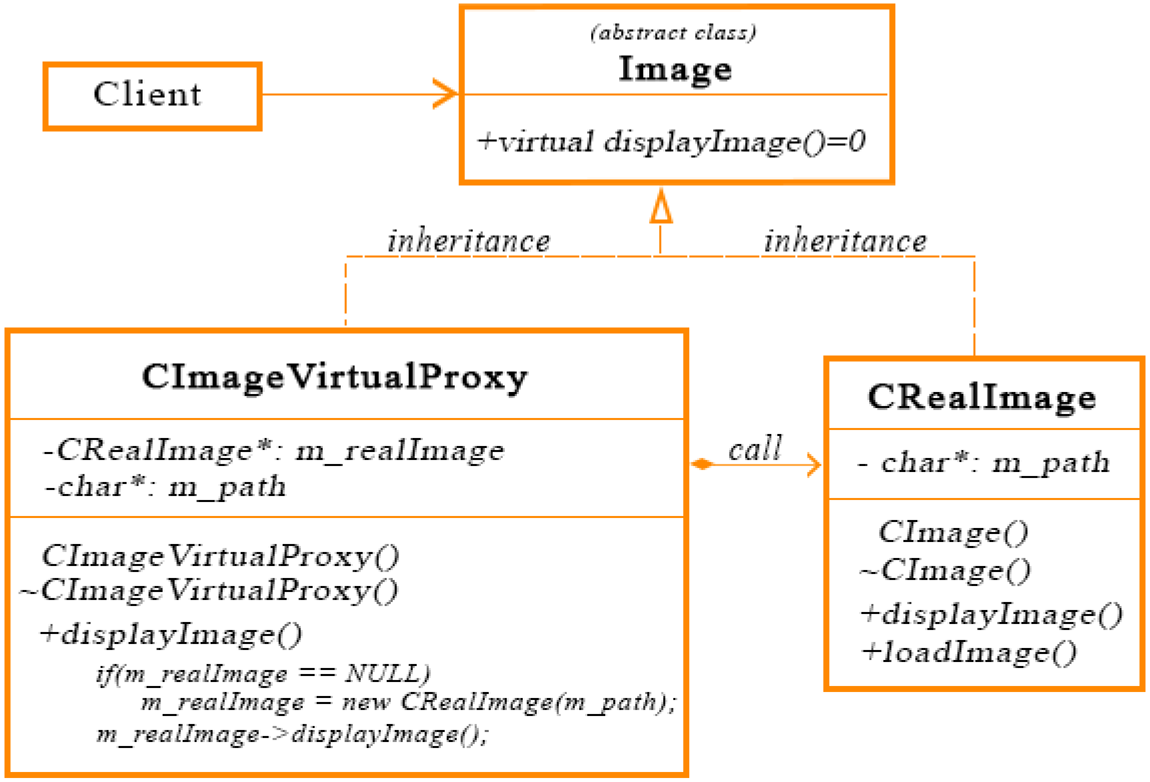
Proxy – class diagram



Proxy – case study

- Virtual proxy

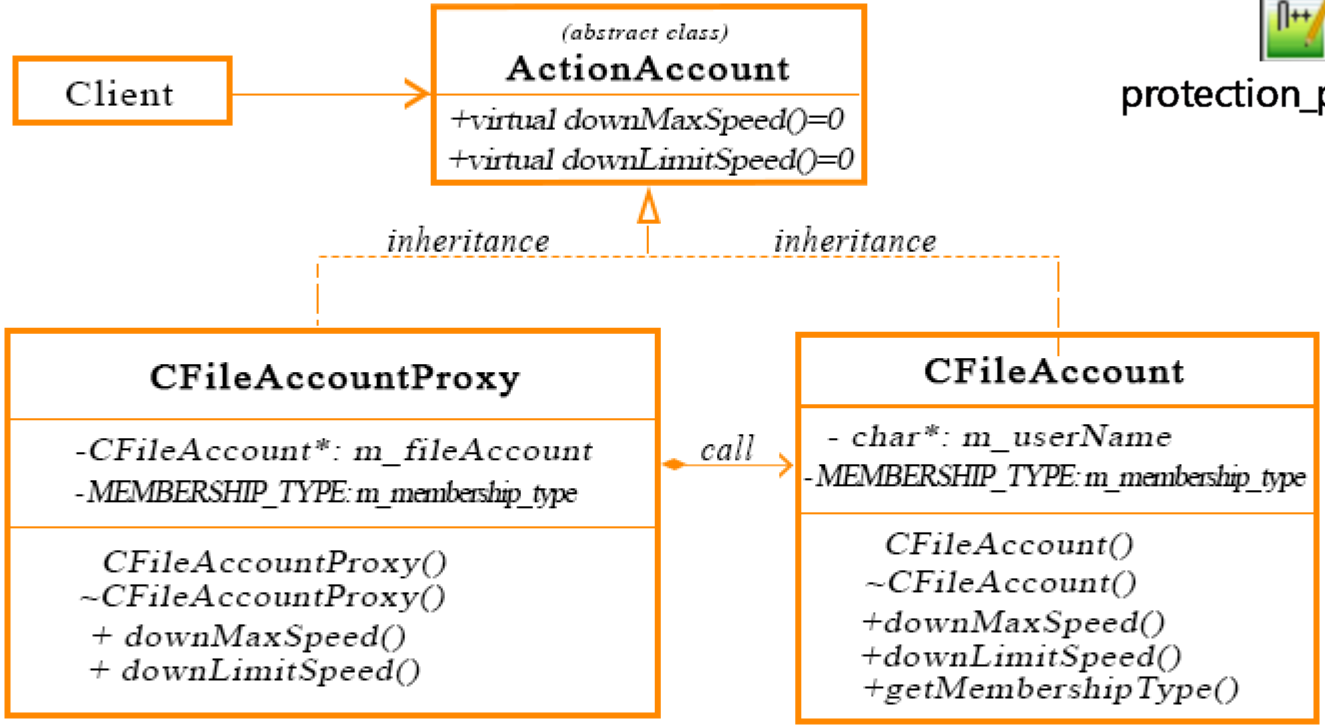
virtual_proxy.cpp



```
if(m_realImage == NULL)
    m_realImage = new CRealImage(m_path);
m_realImage->displayImage();
```

Proxy – case study

- Protection proxy



protection_proxy.cpp

Conclusion

- Don't Get Obsessed With Design Patterns
- Design patterns can be our best ally when used correctly

DCV Discussion

THANKS
ess.
FOR YOUR ATTENTION
memegenerator.net