



Design Patterns in use

Pascal Bihler

Intern NW F AUI Styles & FES, SAP AG

THE BEST-RUN BUSINESSES RUN SAP



Introduction to Design Patterns

Examples: Resources in Eclipse

Other Patterns used by Eclipse

**A Design Pattern describes
a family of solutions
for a given Software-Design problem**

(The Pattern is not the solution itself, but more an idea of a solution!)

**The final goal of Design Patterns is the
reusability of Software Design knowledg**

Why Design Patterns?

- **Design Patterns can help to improve team communication**
- **Patterns document and facilitate the state-of-the-art**
- **Patterns reflect main concepts and can help to understand, clarify and document design decisions**
- **Patterns can help to avoid design drift**
- **Patterns can improve code structure and code quality**

Which Design Patterns exists?

Isolation

Modul

Abstract Data Type: Repository, Client/Server, Manager, Iterator, Collection

Layer-Architecture: Sandwich, Façade, Mediator, **Bridge**, Adapter, Facet, **Proxy** (Decorator, Buffering Proxy, Logging Proxy, Firewall, Synchronizer, Remote Access Proxy)

Data Flow (Pipes and Filters)

Event Handling: Throw/catch, Callback, Event-Loop, Event-Channel, Propagator (Exact, Fuzzy, Lazy, Adaptable, **Observer**)

Framework

Variants

Family: Strategy **Composite**

Visitor (Simple, Outside, Acyclic)

Template Method: Fabric Method, Builder

Abstract Factory

Virtual Machines

Interpreter: Emulator

Rule based Interpreter

State

Memento

Prototype

Flyweight

Singleton

Control

Blackboard

Command

Chain of Responsibility

Strategy

Control state

Master/Slave

Process Control
(with/without feedback)

Convenience

Convenience method

Convenience class

Façade

Null-Object



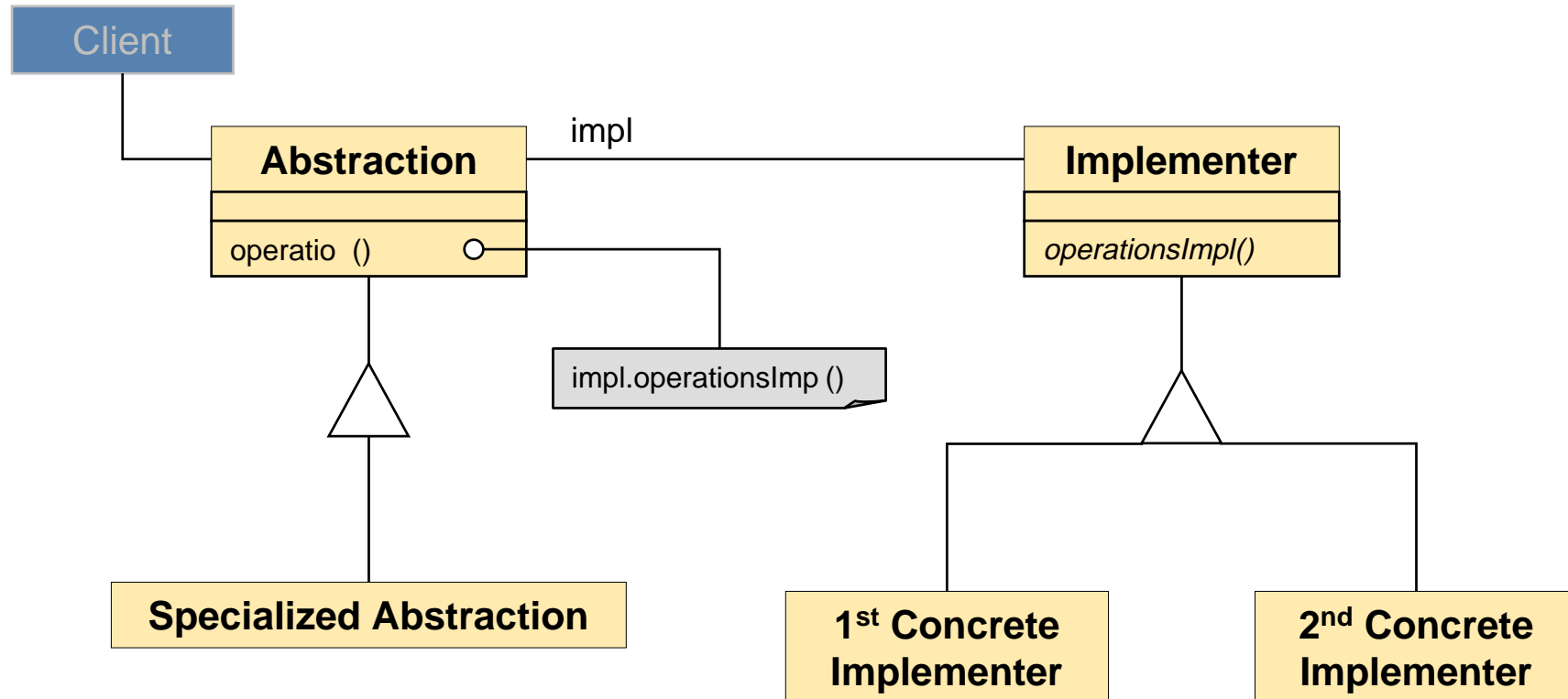
Introduction to Design Patterns

Examples: Resources in Eclipse

Other Patterns used by Eclipse

Bridge (Handle/Body)

⇒ Separation of abstract view and implementation



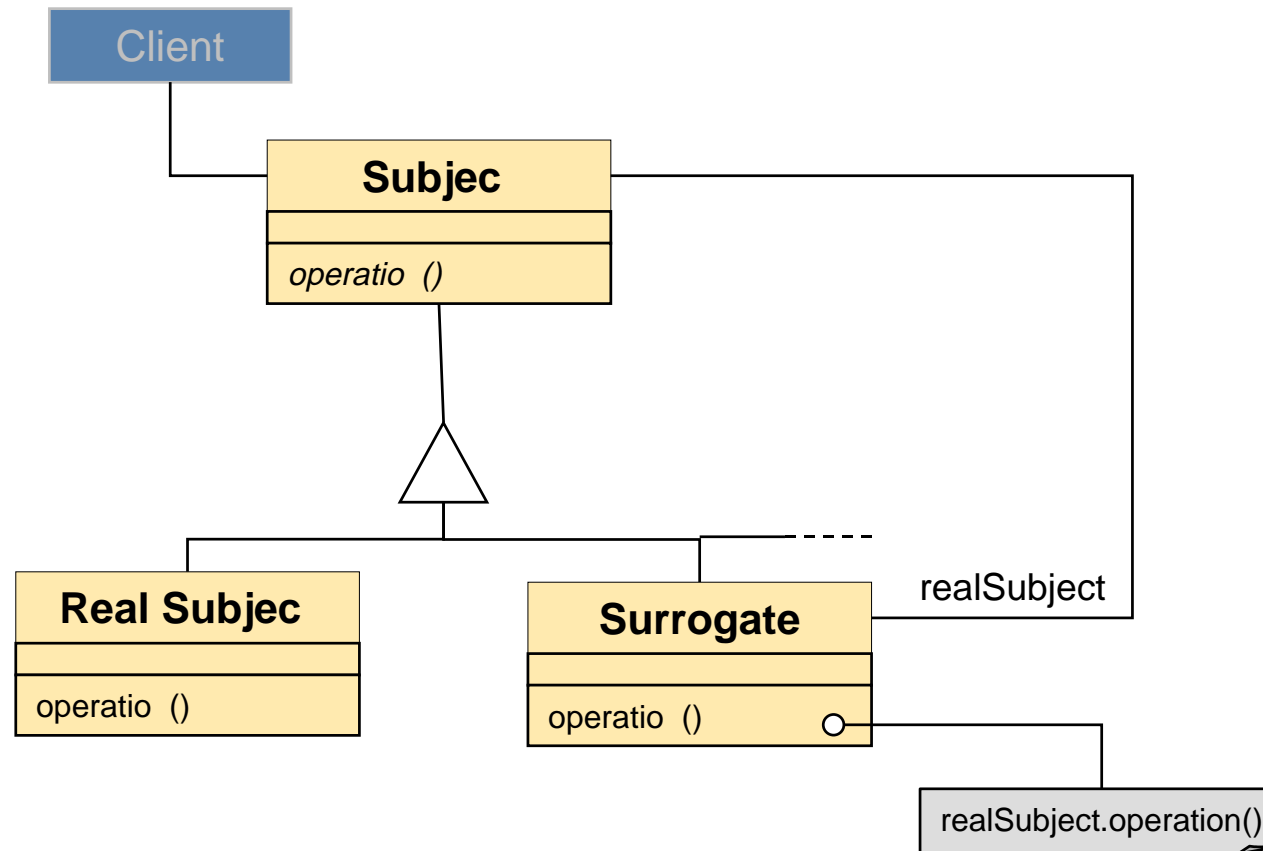
Bridge (Handle/Body)

Use a bridge if you...

- ... want to avoid a permanent connection between the abstraction and the implementation
- ... want to extend the abstraction as well as the implementation individually
- ... want to hide the implementation of an abstraction completely from the client
- ... want to avoid an increasing number of subclasses during evolution of the project
- ... want to use the implementation by more than one object

Proxy (Surrogate)

⇒ Provide a surrogate or place holder for another object to control access on it



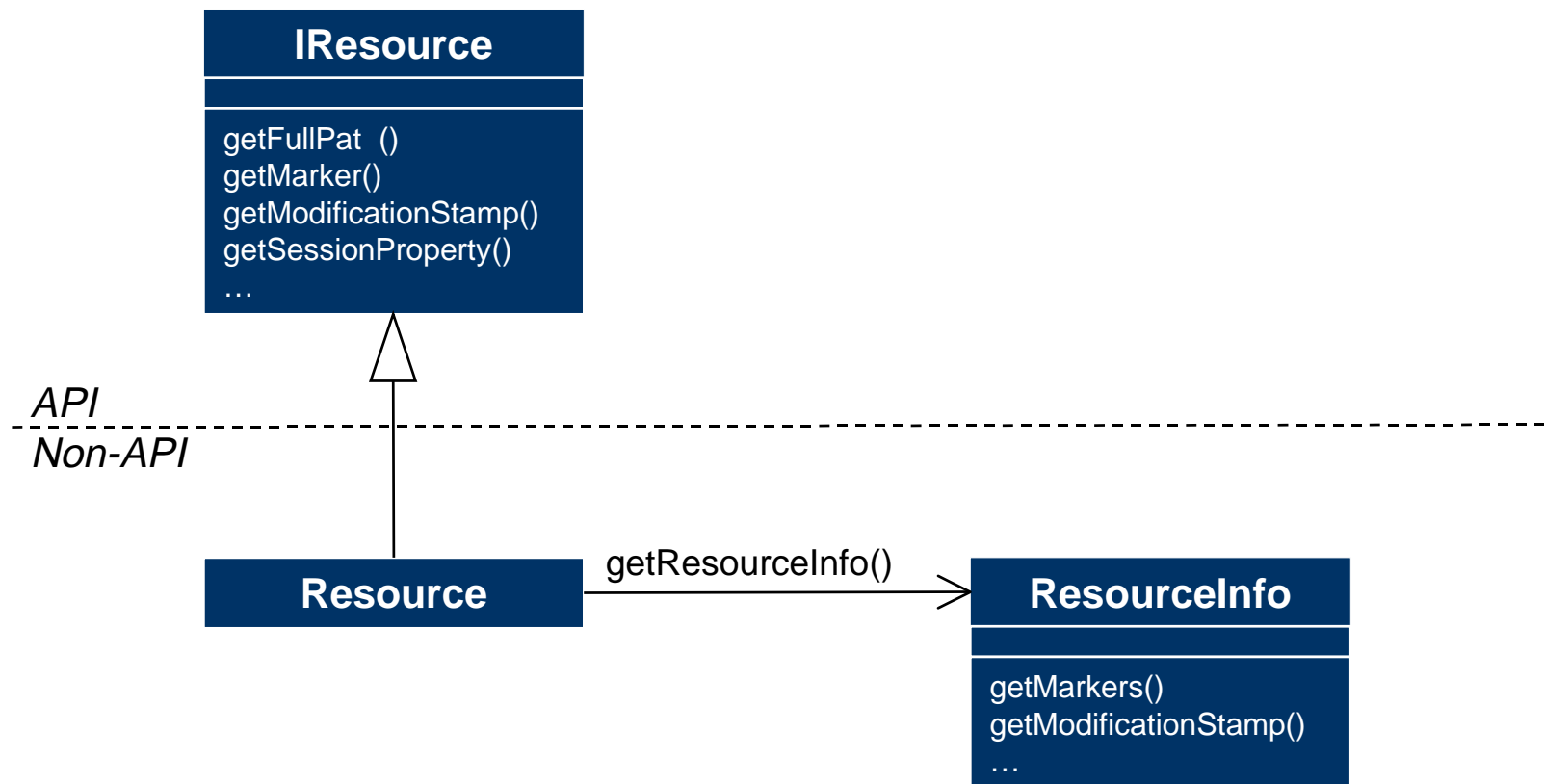
Proxy (Surrogate)

Some forms of proxies (Examples):

- **Logging Proxy:** Reference-counting
- **Buffering Proxy:** Encapsulates memory-movements
- **Remote Access Proxy:** Local access to an object which resides in another address space
- **Surrogate:** Loading the object dynamical during first access
- **Protector:** Controls access on the real object
- **Decorator:** Constitutes additional competence to the object

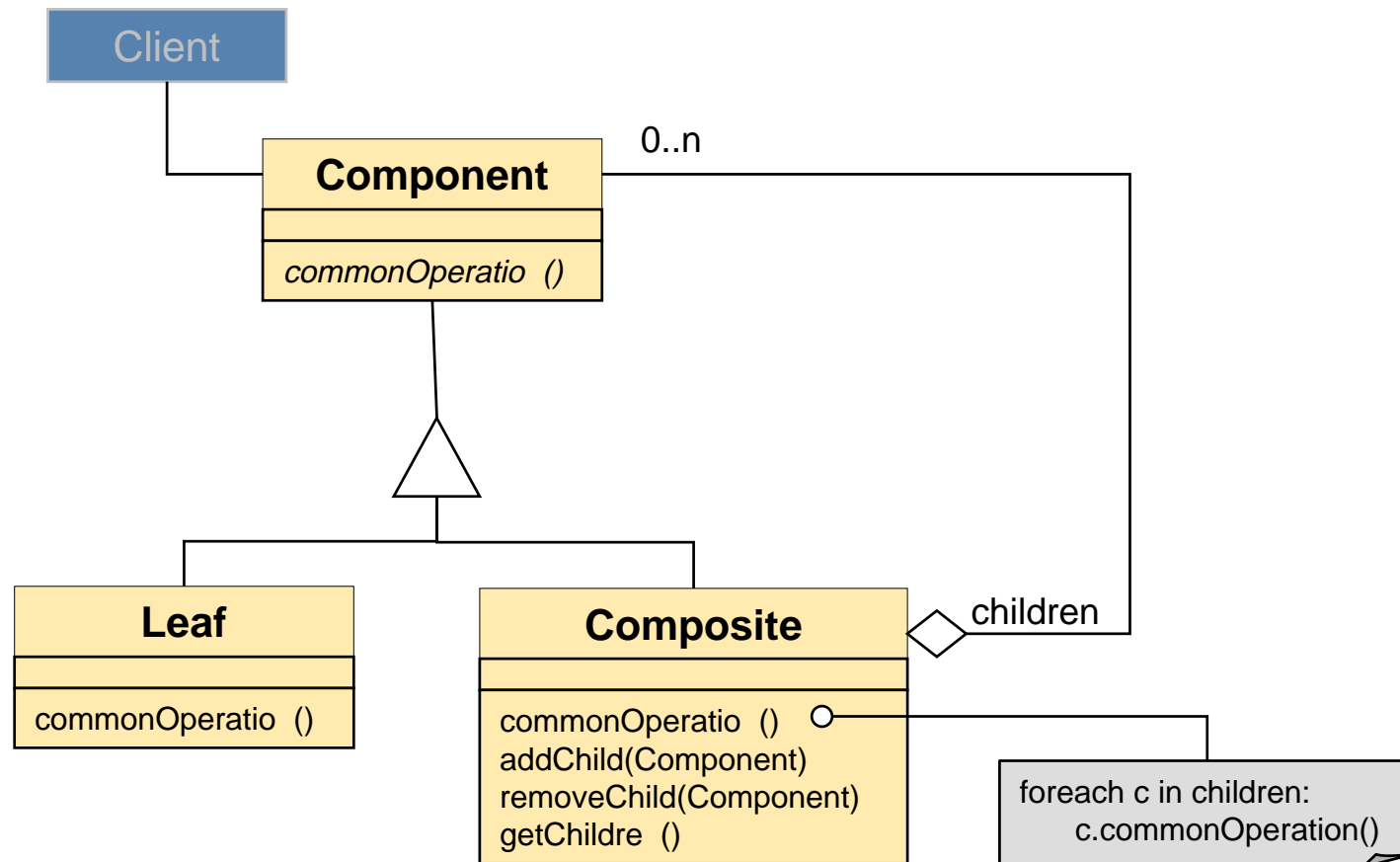
IResource: Bridge and Proxy in Eclipse

The files-system access in Eclipse is provided using proxies and bridges (IFile, IResource, ...)



Composite

⇒ Treat single objects and aggregations of them the same way

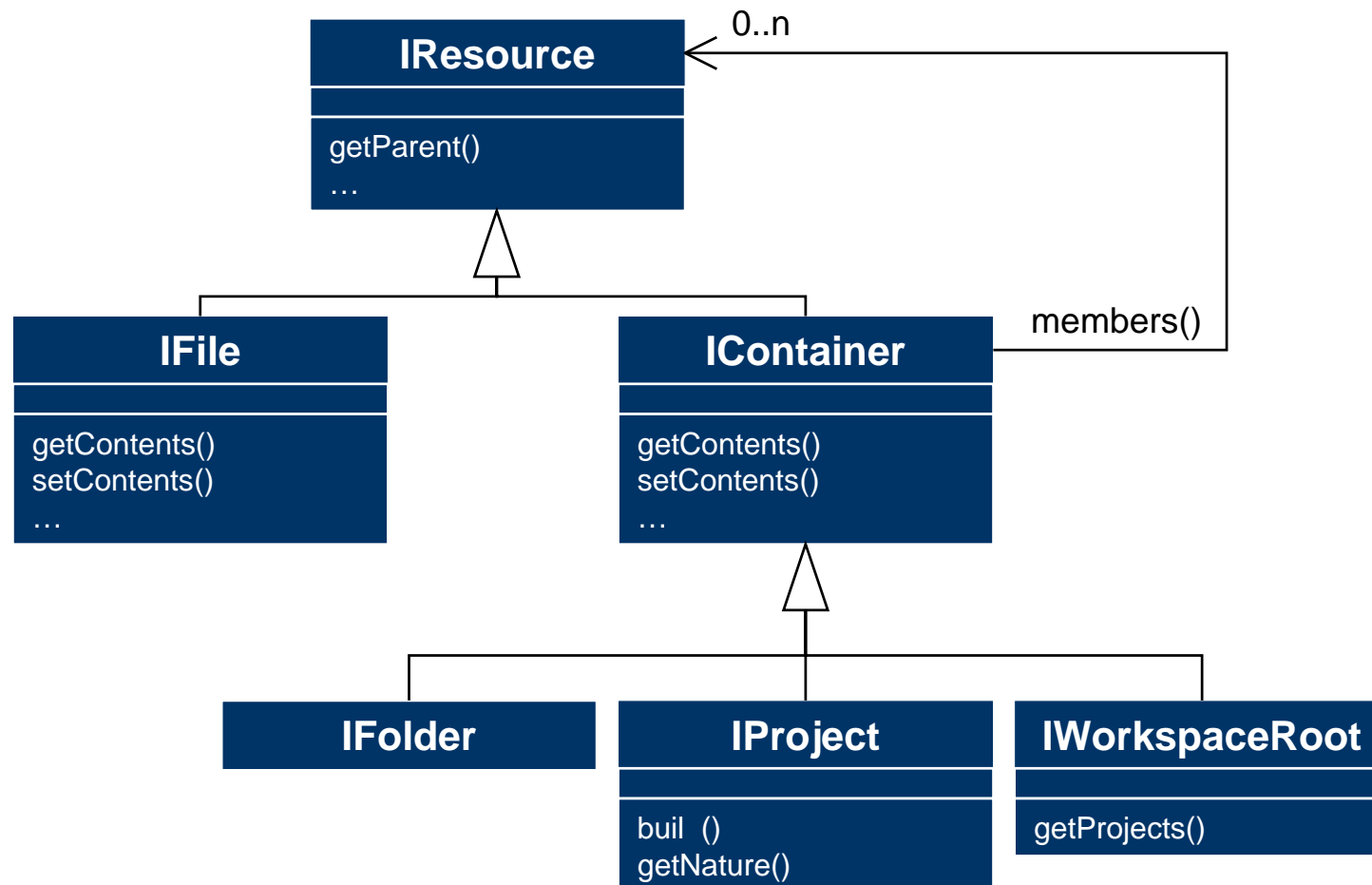


Use a composite if you want to...

- **... represent tree-structures (object hierarchies)**
- **... handle the leafs and aggregates sometimes the same way**
- **... avoid create a special structure for storing the tree-relations**

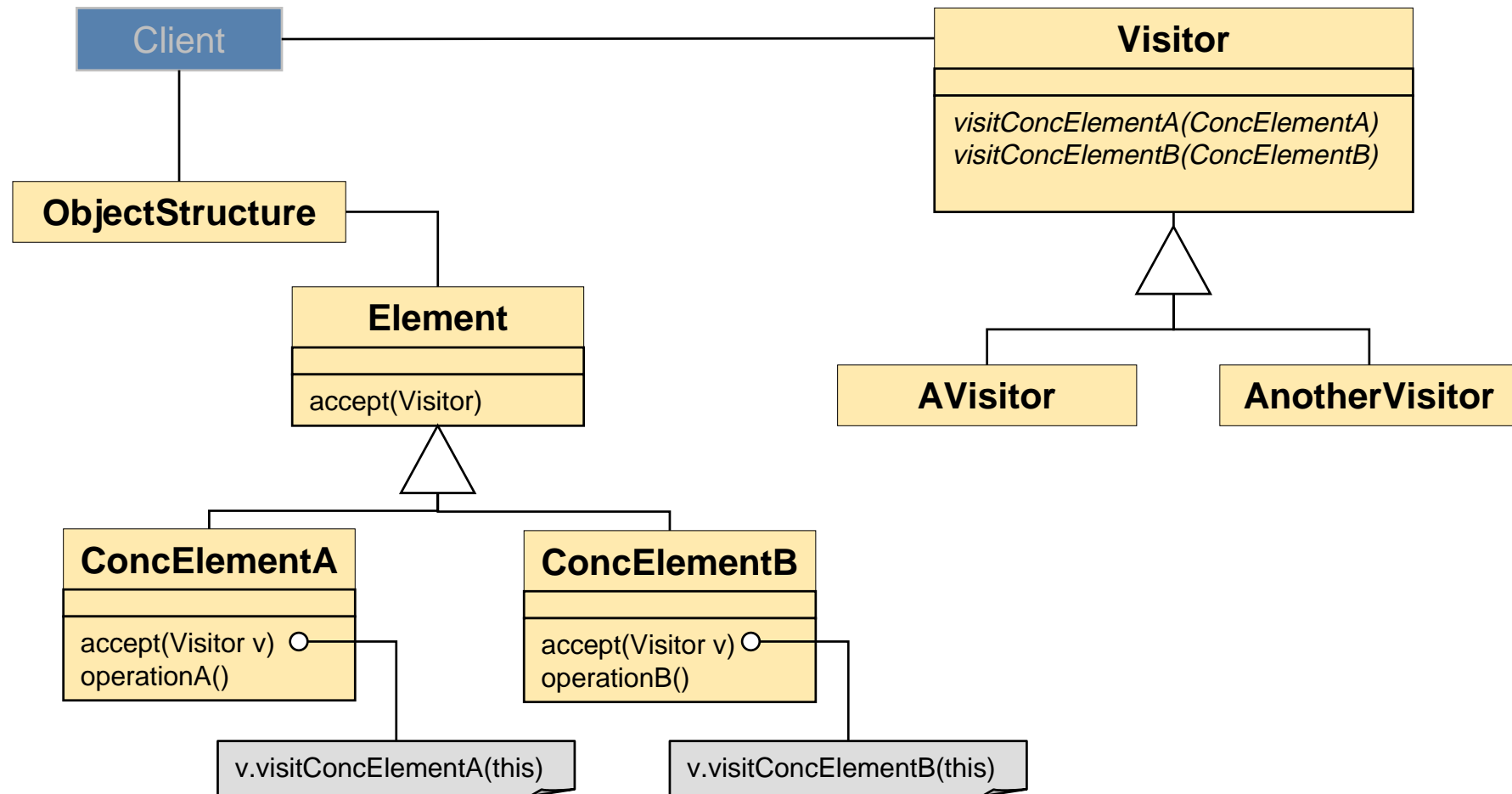
Workspace: Composite

The Eclipse workspace provides resources stored in the file system. This resource tree is implemented using the composite pattern:

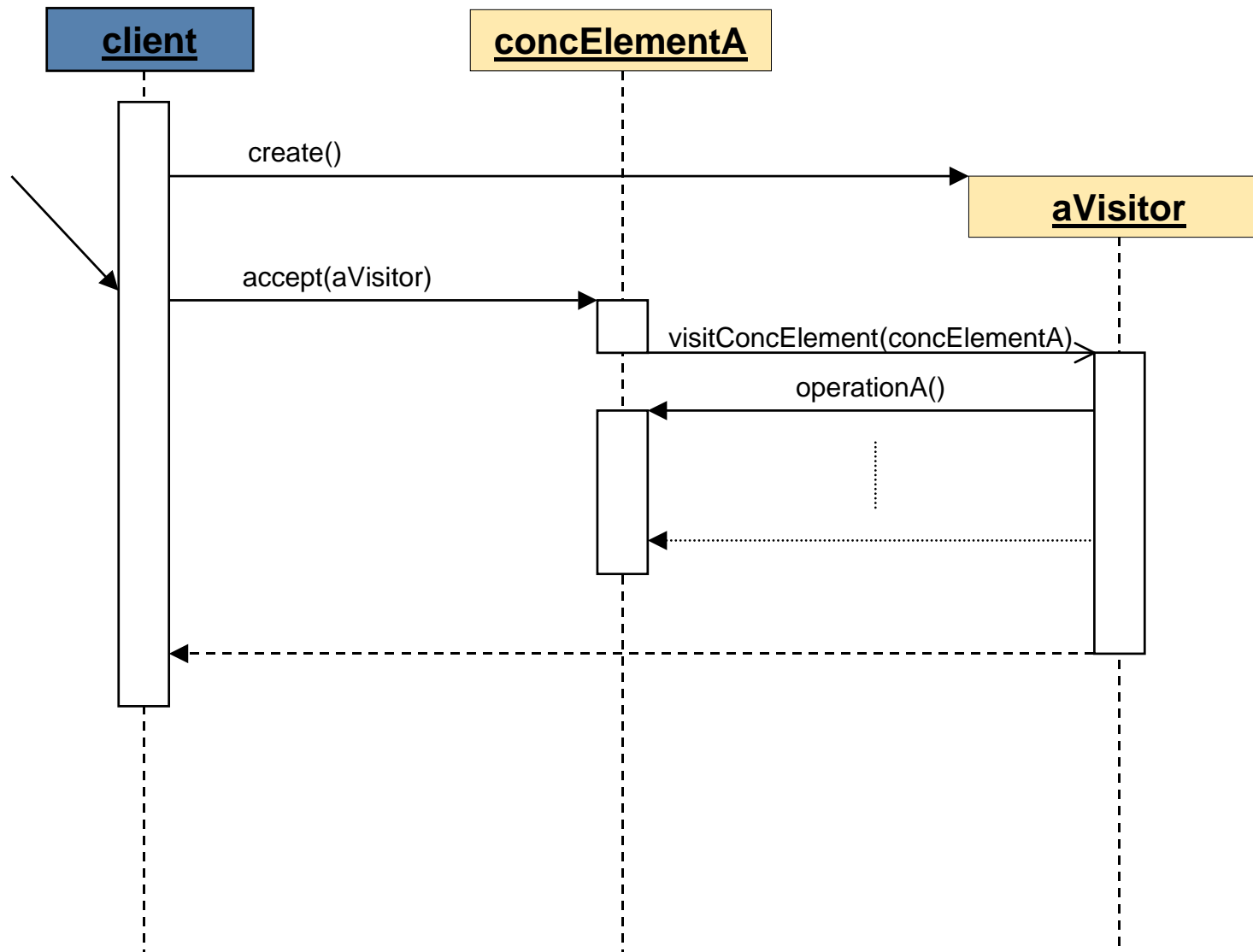


Visitor

⇒ Represents an operation to be performed on the elements of an object structure. Define this operation without changing the classes of the elements on which it operates



Visitor: Sequence Diagramm

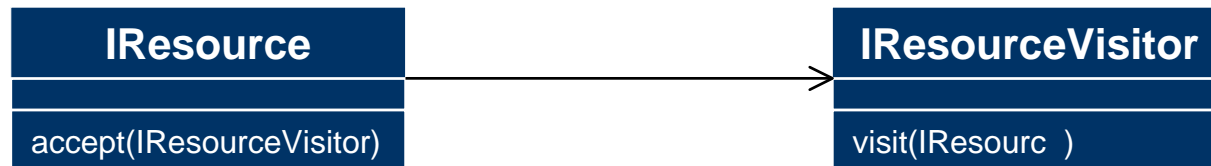


Use a visitor pattern if you want to...

- ... define operations on tree structures containing a lot of elements with different element-interfaces and you have to use these special interfaces
- ... create different and not related operations on an object structure and you don't want to pollute the objects' classes with these operations
- ... define frequently new operations on objects whose interfaces nearly never change

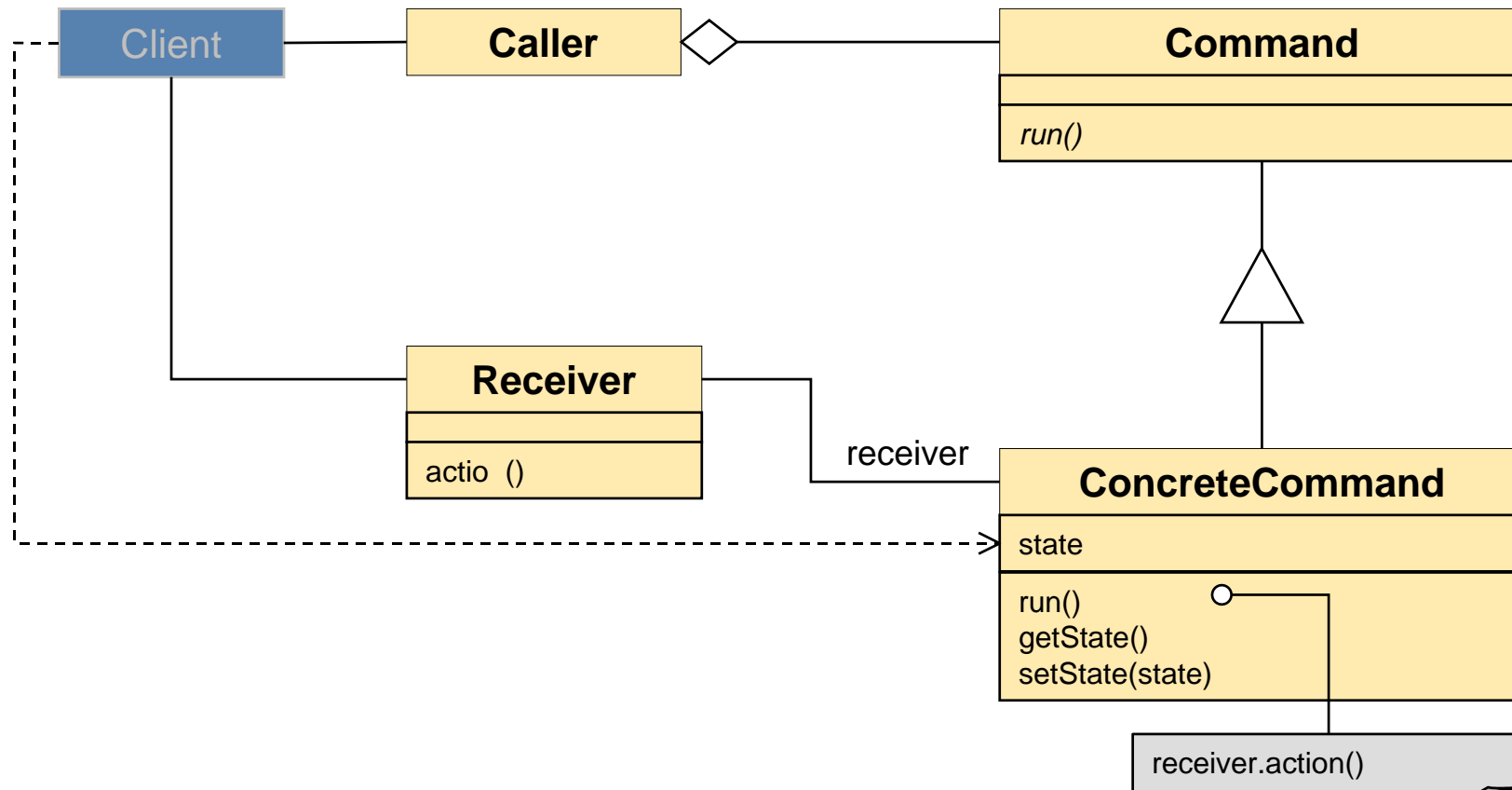
Traversing the Resource Tree: Visitor

The Visitor-Pattern is used in Eclipse to provide operations on the resources in the workspace:



Command

⇒ An operation is encapsulated as an object



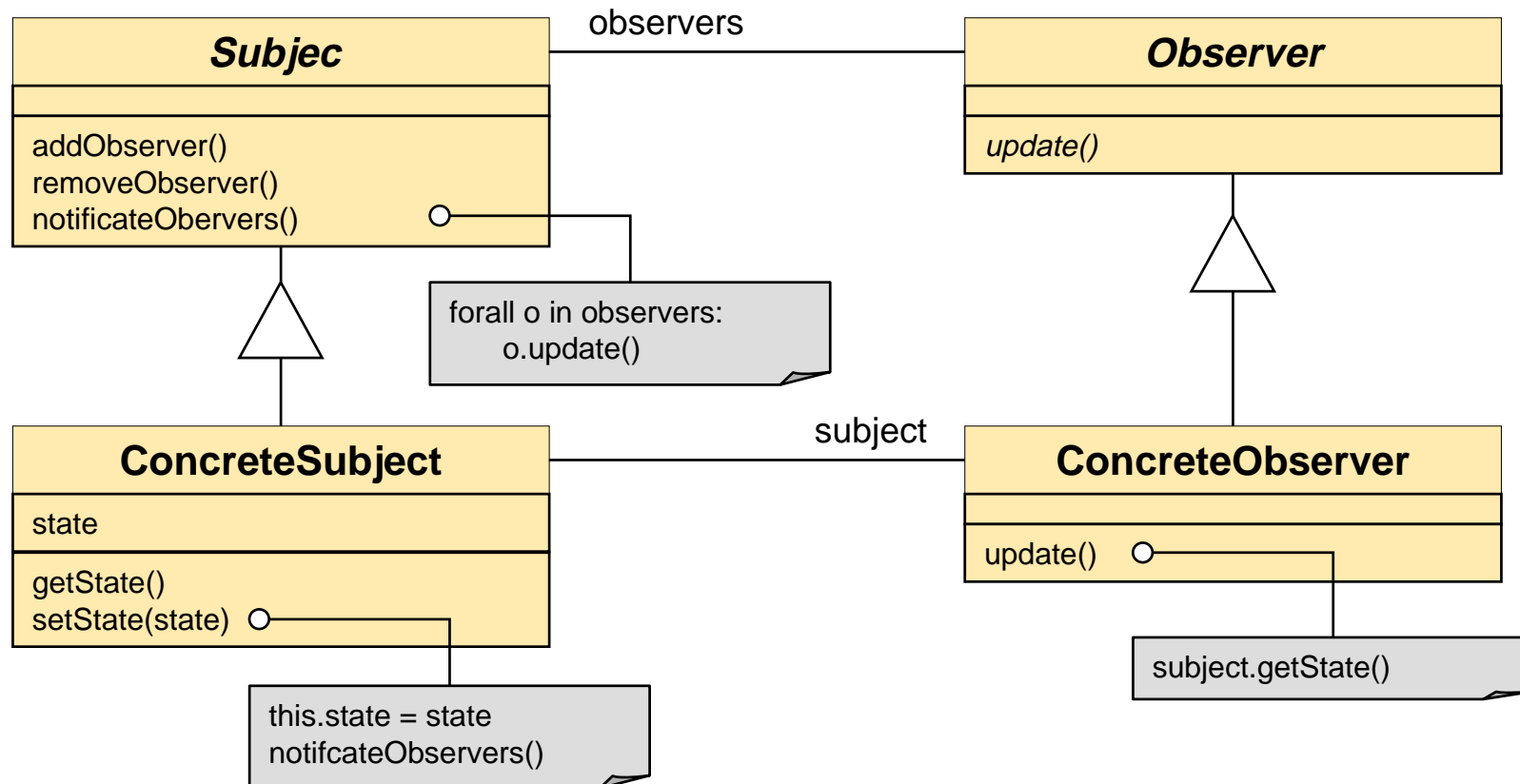
Actions in SWT: Command

In SWT, operations are encapsulated as commands using the **IAction** structure:

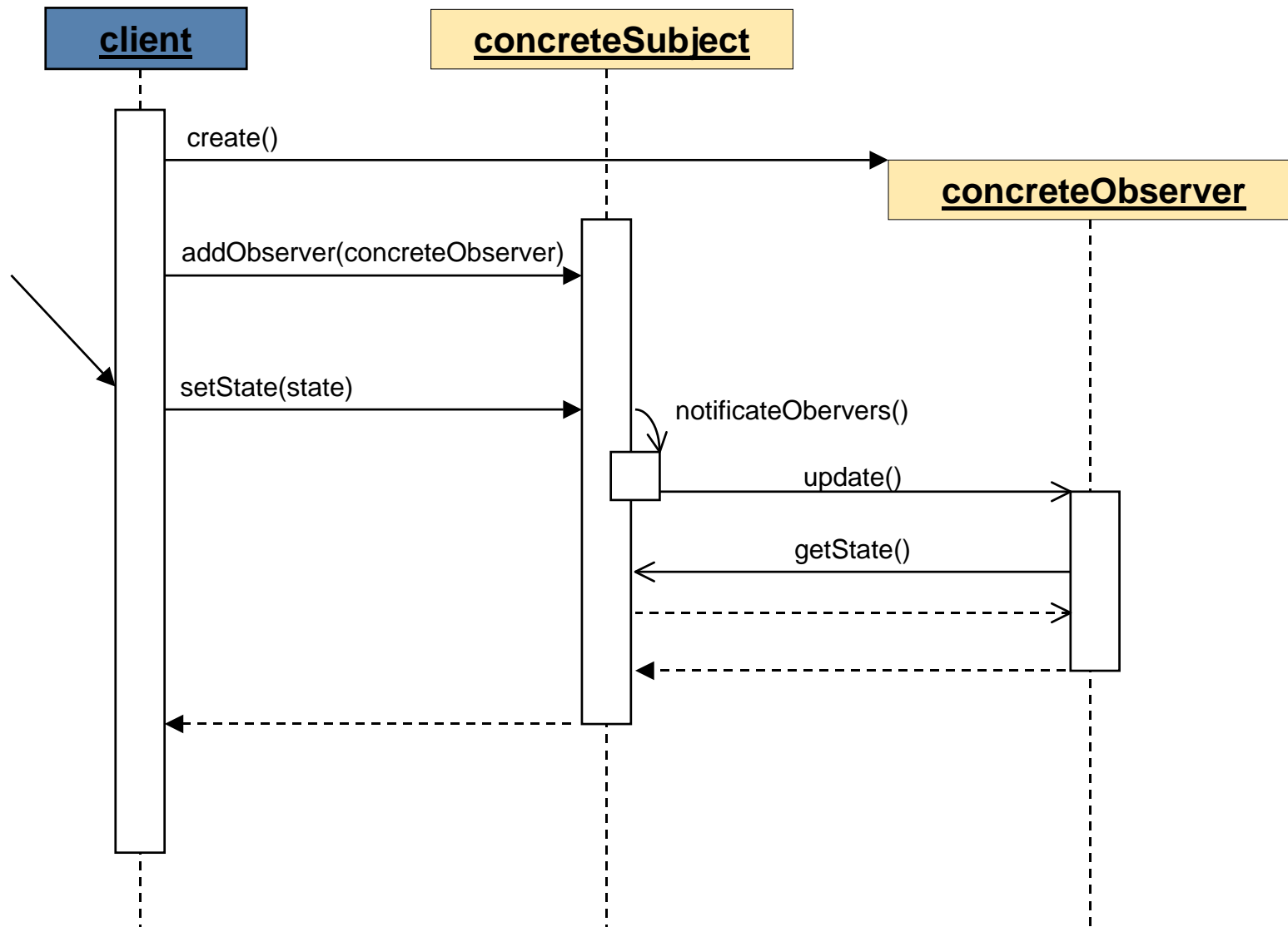
```
IAction  
  
run()  
getText()  
getImageDescriptor()  
getToolTipText()  
isEnabled()  
...
```

Observer

⇒ Define a 1:n connection between objects to inform and update dependent objects when the subject changes



Observer: Sequence Diagramm

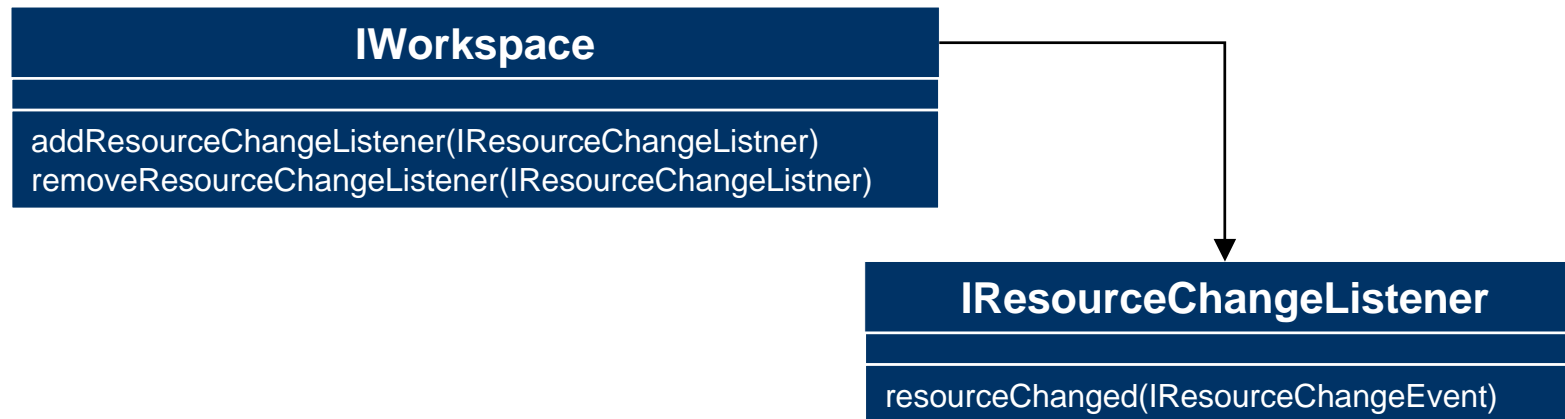


Use an observer if ...

- ... the change of one object effectuate the change of many dependent objects
- ... if an object has to inform other one of it's changes, even if it doesn't know at design time, which and how many objects
- ... if an abstraction contains two aspects which are dependent one from another. By encapsulating them in separate classes, one can easily reuse them.

Tracking Resource Changes: Observer

To track the changes on resources, Eclipse is using the observer-pattern. The observers are called “Listeners”:



SWT’s **IAction** is using the same construction to propagate property changes:



Introduction to Design Patterns

Examples: Resources in Eclipse

Other Patterns used by Eclipse

Other Patterns used in Eclipse

There are a lot of patterns used in Eclipse, which cannot be introduced here separately, e.g.:

- **(Abstract-) Factory:** JavaCore, AdapterFactory
- **Builder:** Collecting JavaCore results
- **Strategy:** Customizing viewers, Projec -Build, SWT-Layouts
- **(Pluggable-) Adapter:** Viewer, IAdaptable, IResource
- **Façade:** Adapter-Manager (IAdapter)
- **Memento:** Restoring UI State
- **Event Channel, Execute Around Method, Virtual Proxy, Prototype, Singleton, Convenience Method/Class, Null-Object,...**



**Thank you very much
for your attention!**

Any questions left?

Used Literature:

- Tichy, Prof. Dr. Walter; *Script for lecture "Softwaretechni "*; Technical University of Karlsruhe, winter term 2003/2004:
<http://www.ipd.uka.de/Tichy/SW>
- Gamma, Erich; Beck, Kent; *Contributing to Eclipse: Principles, Patterns, and Plu -Ins*; Addison Wesley Professional.
- Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John; *Design Patterns*; Addison-Wesley Pub Co

Copyright 2004 SAP AG. All Rights Reserved

- No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.
- Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.
- Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.
- IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.
- Oracle is a registered trademark of Oracle Corporation.
- UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.
- Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.
- HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.
- Java is a registered trademark of Sun Microsystems, Inc.
- JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.
- MaxDB is a trademark of MySQL AB, Sweden.
- SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.
- These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.