

Object Oriented Frameworks (in Work)

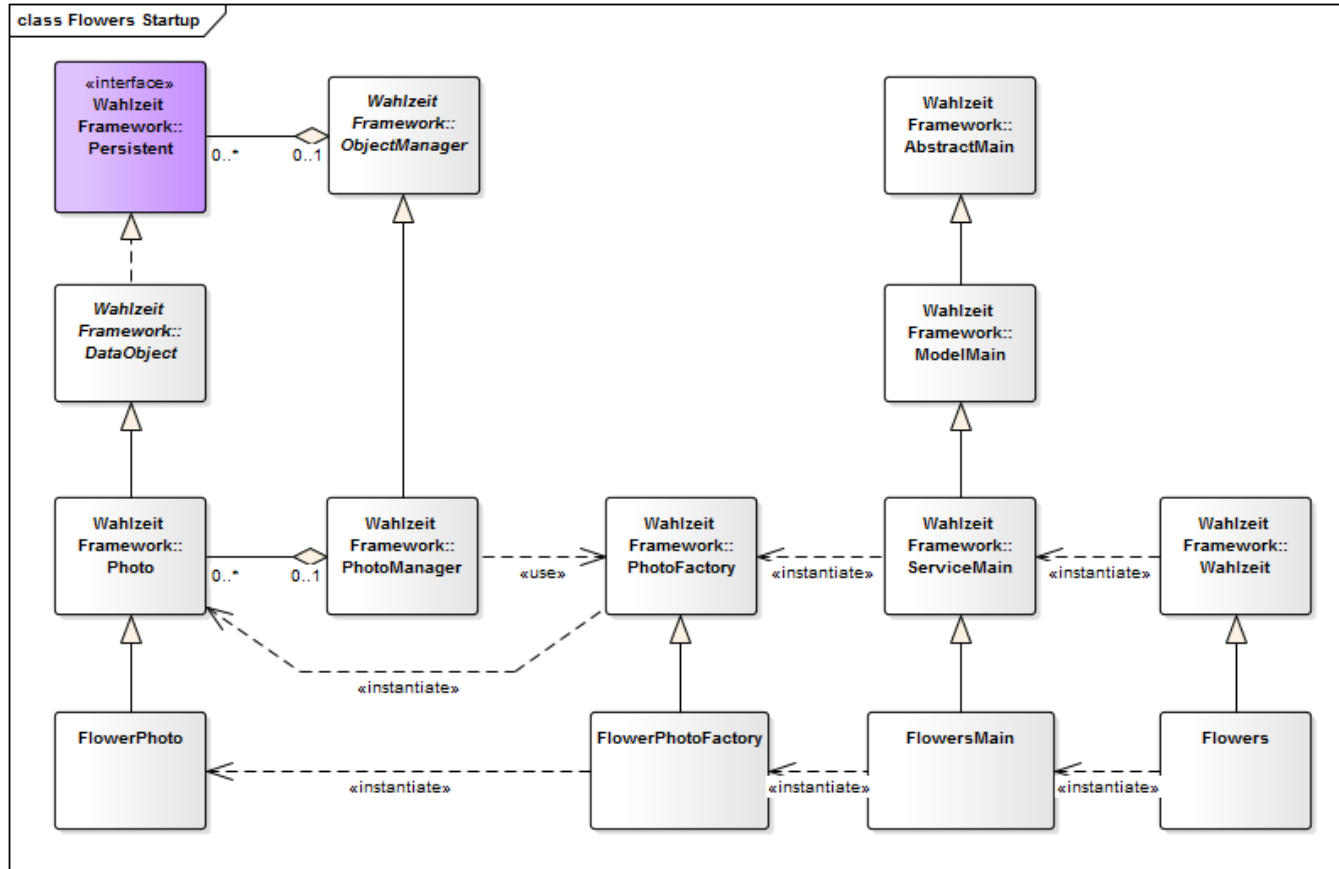
Prof. Dr. Dirk Riehle

Friedrich-Alexander University Erlangen-Nürnberg

ADAP C12

Licensed under [CC BY 4.0 International](https://creativecommons.org/licenses/by/4.0/)

Wahlzeit Start-up



- 1. Definition**
- 2. Interfaces**
- 3. Packaging**
- 4. Layering**

Object-Oriented Framework

- Definition of object-oriented framework
 - Is an abstract object-oriented design that can be reused
 - Has default implementation classes that can be used
 - Typically covers one particular technical domain
- White-box framework
 - An object-oriented framework mostly used by implementing subclasses
 - Requires user to understand internal workings of framework
 - Typically form of a framework in its early stages
- Black-box framework
 - An object-oriented framework mostly used by composing instances
 - Easier to use by less flexible than white-box framework
 - Typically form of a framework in its mature stages

Frameworks vs. Libraries (Toolkits)

- Frameworks

- Reuse of abstract design
 - Inheritance and delegation
 - Inheritance interface
 - High cohesion of classes
 - More difficult to use than library
- Examples
 - Java Object framework
 - Wahlzeit domain model

- Libraries

- No reuse of abstract design
 - No or little use of inheritance
 - Only use-relationship
 - Loose class relationships
 - Easier to use than framework
- Examples
 - Java utility classes
 - Wahlzeit utility classes

- 1. Use-client interface**
- 2. Inheritance interface**
- 3. Meta-object Protocol**

Use-Client Interface

- The use-client interface is the traditional interface
 - Invoked using method calls by client objects on framework objects
- Best practices of defining use-client interfaces
 - An abstract object-oriented design that reflects the domain
 - Using interfaces, abstract classes, and implementation classes
 - Using collaborations spelling out roles and their responsibilities
 - Using exceptions properly to document behavior in case of failure
 - With clear idea of types of objects, for example, value objects
 - With clear idea of patterns employed to structure the design

Inheritance Interface

- The inheritance interface uses polymorphism
 - Subclasses extend the design while conforming to it
 - Leads to inverted control-flow, a.k.a. “Hollywood principle”
- Best practices of defining inheritance interfaces
 - An abstract object-oriented design that reflects the domain
 - Using the abstract superclass rule
 - Using the narrow inheritance interface principle
 - With clear idea of patterns employed to structure the interface, e.g.
 - Primitive and composed methods
 - Factory method, template, method, etc.
 -
 - Document extension points

Meta-object Protocol (Java Annotations)

Framework Packaging

- Frameworks **have** a design but **are** code components
- Package a framework as a code component
 - Group by object collaborations, not class inheritance
 - Utilize language-specific packaging mechanism
- If necessary, separate functional packages
 - Interface, implementation, and testing packages

Framework Layering

- Frameworks use and extend other frameworks
 - Use = utilizing use-client interfaces
 - Extend = utilizing inheritance interfaces
- Use needs to be prepared for, foreseen by designers
 - Utilizing the two general types of interfaces
 - Using specific mechanisms like role objects
- Code layers in an application reflect framework layers

Thank you! Questions?

dirk.riehle@fau.de – <http://osr.cs.fau.de>

dirk@riehle.org – <http://dirkriehle.com> – [@dirkriehle](#)

Credits and License

- Original version
 - © 2012-2019 [Dirk Riehle](#), some rights reserved
 - Licensed under [Creative Commons Attribution 4.0 International License](#)
- Contributions
 - ...

Object Oriented Frameworks (in Work)

Prof. Dr. Dirk Riehle

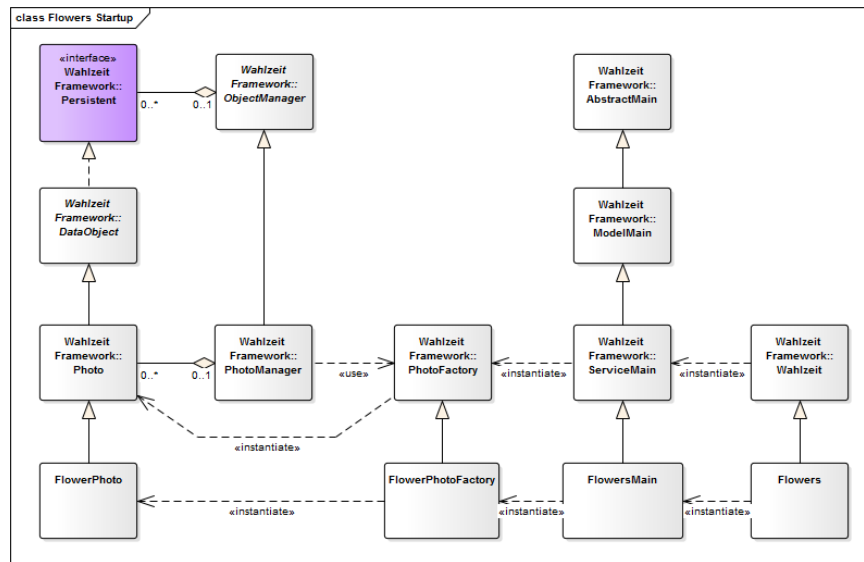
Friedrich-Alexander University Erlangen-Nürnberg

ADAP C12

Licensed under [CC BY 4.0 International](#)

It is Friedrich-Alexander University Erlangen-Nürnberg – FAU, in short.
Corporate identity wants us to say “Friedrich-Alexander University”.

Wahlzeit Start-up



- 1. Definition**
- 2. Interfaces**
- 3. Packaging**
- 4. Layering**

Object-Oriented Framework

- Definition of object-oriented framework
 - Is an abstract object-oriented design that can be reused
 - Has default implementation classes that can be used
 - Typically covers one particular technical domain
- White-box framework
 - An object-oriented framework mostly used by implementing subclasses
 - Requires user to understand internal workings of framework
 - Typically form of a framework in its early stages
- Black-box framework
 - An object-oriented framework mostly used by composing instances
 - Easier to use by less flexible than white-box framework
 - Typically form of a framework in its mature stages

Frameworks vs. Libraries (Toolkits)

- Frameworks
 - Reuse of abstract design
 - Inheritance and delegation
 - Inheritance interface
 - High cohesion of classes
 - More difficult to use than library
 - Examples
 - Java Object framework
 - Wahlzeit domain model
- Libraries
 - No reuse of abstract design
 - No or little use of inheritance
 - Only use-relationship
 - Loose class relationships
 - Easier to use than framework
 - Examples
 - Java utility classes
 - Wahlzeit utility classes

- 1. Use-client interface**
- 2. Inheritance interface**
- 3. Meta-object Protocol**

Use-Client Interface

- The use-client interface is the traditional interface
 - Invoked using method calls by client objects on framework objects
- Best practices of defining use-client interfaces
 - An abstract object-oriented design that reflects the domain
 - Using interfaces, abstract classes, and implementation classes
 - Using collaborations spelling out roles and their responsibilities
 - Using exceptions properly to document behavior in case of failure
 - With clear idea of types of objects, for example, value objects
 - With clear idea of patterns employed to structure the design

Inheritance Interface

- The inheritance interface uses polymorphism
 - Subclasses extend the design while conforming to it
 - Leads to inverted control-flow, a.k.a. “Hollywood principle”
- Best practices of defining inheritance interfaces
 - An abstract object-oriented design that reflects the domain
 - Using the abstract superclass rule
 - Using the narrow inheritance interface principle
 - With clear idea of patterns employed to structure the interface, e.g.
 - Primitive and composed methods
 - Factory method, template, method, etc.
 - Document extension points

Meta-object Protocol (Java Annotations)

Framework Packaging

- Frameworks **have** a design but **are** code components
- Package a framework as a code component
 - Group by object collaborations, not class inheritance
 - Utilize language-specific packaging mechanism
- If necessary, separate functional packages
 - Interface, implementation, and testing packages

Framework Layering

- Frameworks use and extend other frameworks
 - Use = utilizing use-client interfaces
 - Extend = utilizing inheritance interfaces
- Use needs to be prepared for, foreseen by designers
 - Utilizing the two general types of interfaces
 - Using specific mechanisms like role objects
- Code layers in an application reflect framework layers

Thank you! Questions?

dirk.riehle@fau.de – <http://osr.cs.fau.de>

dirk@riehle.org – <http://dirkriehle.com> – [@dirkriehle](#)

DR

Credits and License

- Original version
 - © 2012-2019 [Dirk Riehle](#), some rights reserved
 - Licensed under [Creative Commons Attribution 4.0 International License](#)
- Contributions
 - ...