

# Course Schedule and Homework

**Prof. Dr. Dirk Riehle**

**Friedrich-Alexander University Erlangen-Nürnberg**

**ADAP A02**

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

# Course Schedule

| #  | Class Content                | Homework Due                       |
|----|------------------------------|------------------------------------|
| 01 | Introduction                 | -                                  |
| 02 | Containerization             | Wahlzeit instance                  |
| 03 | Application testing          | Containerized instance             |
| 04 | Method types and properties  | Example tests                      |
| 05 | Class and interface design   | Domain classes                     |
| 06 | Subtyping and inheritance    | Implementation classes             |
| 07 | Design by contract           | Abstract superclasses              |
| 08 | Error and exception handling | Classes with contracts             |
| 09 | Value objects                | Classes with error handling        |
| 10 | Design patterns              | Value object applied               |
| 11 | Type objects                 | Design patterns applied            |
| 12 | Object creation              | Type object applied                |
| 13 | Collaboration-based design   | Object creation applied            |
| 14 | Object-oriented frameworks   | Collaboration-based design applied |
| 15 | Guest talk and conclusions   | Frameworks applied                 |

# Types of Deliverables

- Summary (of homework)
  - Explains your programming homework
  - Should have the following header (at max. ½ page):
    - **Project name**
      - E.g. Flowers
    - **Project repository**
      - E.g. <http://github.com/dirkriehle/wahlzeit>
    - **Project's continuous integration information**
      - E.g. <https://travis-ci.org/andreas-bauer/wahlzeit>
    - **This week's tag**
      - E.g. adap-cw03 on master
    - **This week's diff:**
      - E.g. <https://github.com/dirkriehle/repository/compare/adap-cw02...dirkriehle:adap-cw03>
  - Followed by the explanation of what you did and why (at max. 1.5 page)
  - Name PDF using **lastname-firstname.pdf** convention
  - Includes tagged version of program code

# Time of Day for Weekly Deliverables

- Right before class
  - Submit this week's deliverables

# Plagiarism

- Plagiarism
  - is providing as your work the work of others
  - with or without those other people's consent
- Other people
  - Other people may be other students in the course
  - May be any web source, including Wikipedia
- Citing sources
  - You can cite or use others if you clearly list the source
  - In most cases, you should limit any verbatim use

# Handling Plagiarism

- Detecting plagiarism
  - We review your work and will detect plagiarism
  - Students are required to report any plagiarism they see
- Handling plagiarism
  - Any plagiarizing student will receive 0 points for their work
  - We will review any involved student's full submission history
- Review components
  - For reviews, it is acceptable to reuse your own text components
  - This is called self-plagiarism (and usually not acceptable)

# Open Source License Compliance

- You can copy other people's code to solve some homework if
  - You copy the code only after the homework deadline (to catch-up) and
  - Comply with the license requirements of that code
- Wahlzeit is licensed under the AGPLv3 license
  - All contributed code is thereby also licensed under AGPLv3
  - Combining such code is possible if you comply with the license
- The AGPLv3 license requirements are to
  - Publish your own changes to the source code (done automatically) and to
  - Attribute the original author of the code you use and some more
- Please see <https://www.gnu.org/licenses/agpl-3.0.en.html>

# CW #01 Homework Due

- None



# CW #01 Class Preparation Due

- None

# CW #02 Homework Due

- **This week's required content**
  - **Set-up accounts**
    - GitHub
    - DockerHub
    - Travis CI
  - **Fork Wahlzeit**
- **Set-up environment**
  - **Install necessary applications**
- **Build and start application**
  - **Add at least three photos for illustration**
- **Commit and tag with **adap-hw01** on GitHub**
- **Submit this week's summary**

# CW #02 Class Preparation Due

- **Class preparation (in recommended order)**

- **Required reading**

- [B02a] Beck, K. (2002). The Money Example. Part 1 in [B02].
    - [W98] Woolf, B. (1998). Null object. Chapter 1 in [M+98].

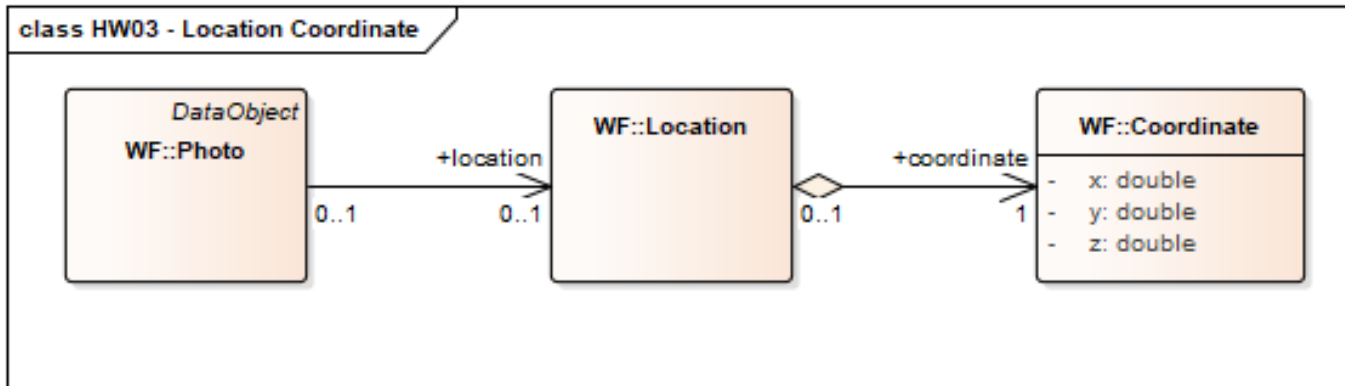
- **Suggested reading**

- [BG98] Beck, K., Gamma, E. (1998). Test infected: Programmers love writing tests. Self-published.
    - [B02] Kent, B. (2002). Test driven development: By example. Addison-Wesley Professional.
    - [V15] Van Deursen, A. (2015). Testing web applications with state objects. Communications of the ACM, 58(8), 36-43.

- This week's **required content**
  - Make project decision (type of photo)
  - Add a Location and Coordinate class to Wahlzeit
  - Containerized application (dockerfile)
- Build, commit, push, and tag with adap-hw02 to GitHub and DockerHub
- Submit this week's summary
  - Use <https://github.com/user/repository/compare/adap-hw01...user:adap-hw02>

## CW #03 Homework Due 2 / 2

- Add a Location and a Coordinate class to Wahlzeit
  - Associate classes as shown in class model
  - Use Cartesian coordinates for Coordinate class
- Implement at least the following methods
  - `double Coordinate#getDistance(Coordinate)` // direct Cartesian distance
  - `boolean Coordinate#isEqual(Coordinate)`
- Forward `equals()` to `isEqual()`



# CW #03 Class Preparation Due

- **Class preparation (in recommended order)**
  - **Required reading**
    - [M07a] Meszaros, G. (2007). Test smells. Chapter 2 in [M07].
    - [M07b] Meszaros, G. (2007). Persistent fixture management. Chapter 9 in [M07].
    - [M07c] Meszaros, G. (2007). Testing with databases. Chapter 13 in [M07].
  - **Suggested reading**
    - [M07] Meszaros, G. (2007). xUnit test patterns: Refactoring test code. Pearson Education.
    - [F04] Feathers, M. (2004). Working effectively with legacy code. Prentice Hall Professional.

# CW #04 Homework Due

- This week's **required content**
  - **Add a new test case to EmailAddressTest**
  - **Add a new test case to EmailServiceTest**
  - Create an EmailService test suite for testing the email service
  - Integrate all test cases into an overall Wahlzeit test suite "AllTests.java"
    - Adjust Gradle test task in build.gradle file, so that "AllTests.java" will be used instead of the wildcard filter
- Build, commit, push, and tag with adap-hw03 to GitHub and DockerHub
- Submit this week's summary

# CW #04 Class Preparation Due

- **Class preparation (in recommended order)**
  - **Required reading**
    - None
  - **Suggested reading**
    - [R00a] Riehle, D. (2000). "Method Types." Java Report 5, 2.
    - [R00b] Riehle, D. (2000). "Method Properties." Java Report 5, 5.
    - [B97a] Beck, K. (1997). Behavior. Chapter 3 in [B97].

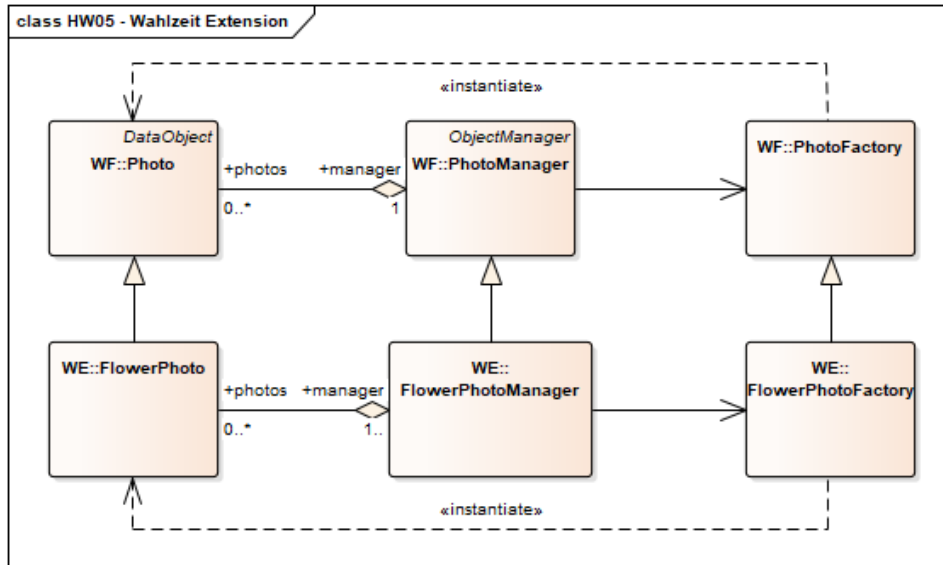


# CW #05 Homework Due 1 / 2

- This week's **required content**
  - Extend Wahlzeit with your Photo class; add others if necessary
  - **Keep adding and adjusting test cases**
- Build, commit, push, and tag
- Submit this week's summary

# CW #05 Homework Due 2 / 2

- Extend Wahlzeit with your Photo class (e.g. FlowerPhoto)
  - Add other classes, where necessary
  - Ensure that your classes plays well with Wahlzeit
    - Specifically, make sure that your photo class is instantiated
    - Make sure that your photos can be saved and loaded using your photo class
- Ignore the user interface (if your class adds new attributes)



# CW #05 Class Preparation Due

- **Class preparation (in recommended order)**

- **Required reading**

- [F97a] Fowler, M. (1997). Class diagrams: The essentials. Chapter 4 in [F97].
- [F97b] Fowler, M. (1997). Class diagrams: Advanced concepts. Chapter 5 in [F97].

- **Suggested reading**

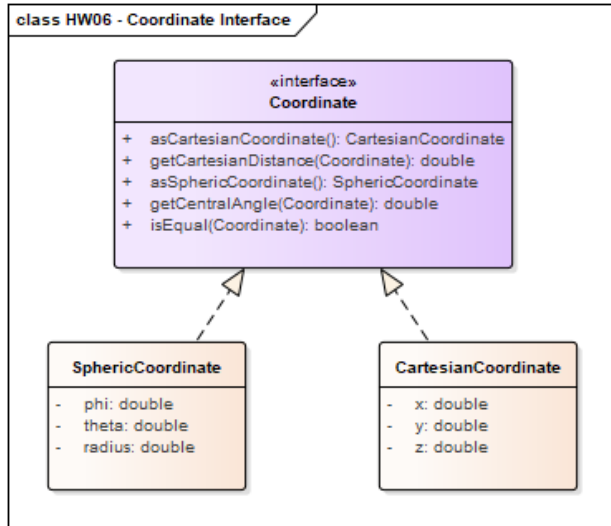
- [RD99a] Riehle, D., & Dubach, E. (1999). Working with Java interfaces and classes. Java Report 4(7).
- [RD99b] Riehle, D., & Dubach, E. (1999). Working with Java interfaces and classes, part 2. Java Report 4(10).

# CW #06 Homework Due 1 / 2

- This week's **required content**
  - Introduce an interface for the **Coordinate** class
  - Add an alternative implementation for the **Coordinate** class
- Build, commit, push, and tag
- Submit this week's summary

# CW #06 Homework Due 2 / 2

- Add a Coordinate interface to Wahlzeit
  - Add an alternative implementation of your current Coordinate class
    - Provide spheric and Cartesian coordinate implementations of Coordinate
    - Ensure that spheric and Cartesian coordinates can be used interchangeably
    - Try to solve it with short methods and no typecasts (but interpretation methods)
    - See [https://en.wikipedia.org/wiki/Great-circle\\_distance](https://en.wikipedia.org/wiki/Great-circle_distance) for definition of central angle
  - Refactor classes accordingly



# CW #06 Class Preparation Due

- **Class preparation (in recommended order)**

- **Required reading**

- [H94] Hürsch, W. L. (1994). Should superclasses be abstract?. In Object-Oriented Programming (pp. 12-31). Springer Berlin Heidelberg.

- **Suggested reading**

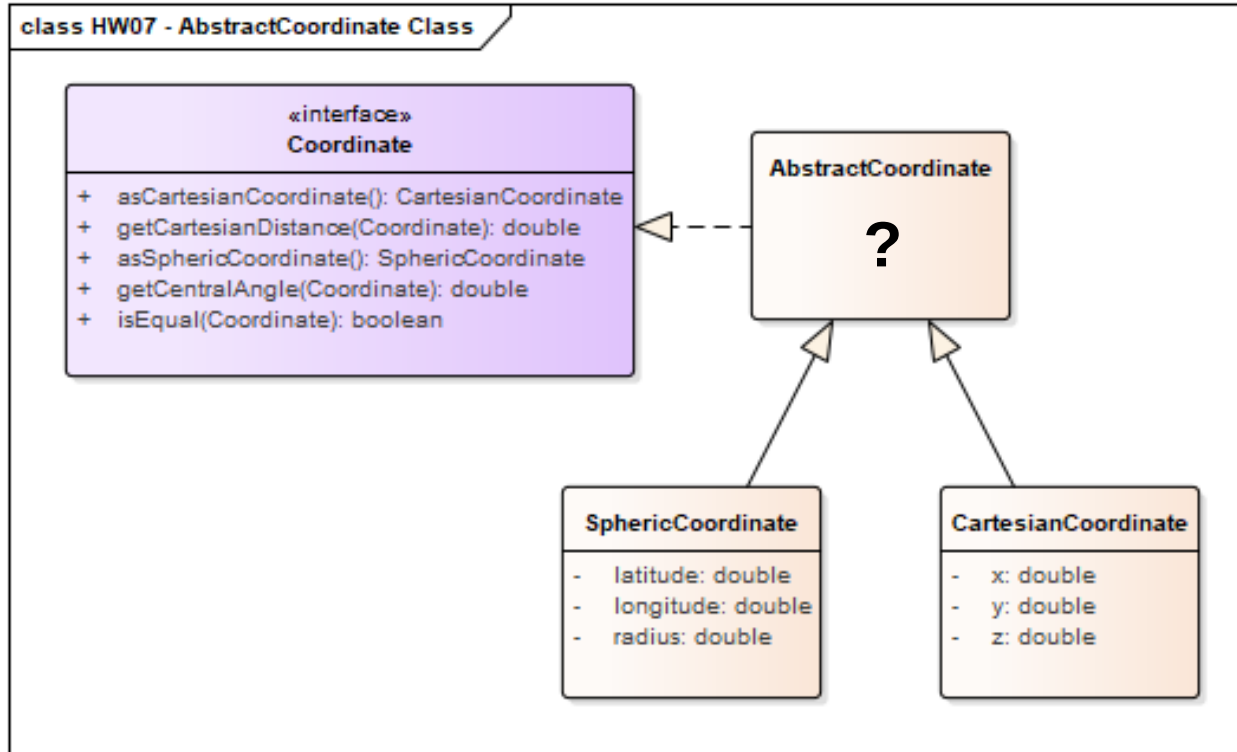
- [LW94] Liskov, B. H., & Wing, J. M. (1994). A behavioral notion of subtyping. ACM Transactions on Programming Languages and Systems, 16(6), 1811-1841.
    - [G+95f] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Template Method. Pattern in [G+95].

# CW #07 Homework Due 1 / 2

- This week's **required content**
  - Introduce an abstract superclass for the **Coordinate** class hierarchy
- Build, commit, push, and tag
- Submit this week's summary

# CW #07 Homework Due 2 / 2

- Introduce an abstract superclass AbstractCoordinate
  - Refactor for minimal redundancy using an inheritance interface





# CW #07 Class Preparation Due

- **Class preparation (in recommended order)**
  - **Required reading**
    - [M91] Meyer, B. (1991). Design by contract. Chapter 1 in [MM91].
  - **Suggested reading**
    - [DS10] Dietrich, J., & Stewart, L. (2010). Component contracts in eclipse-a case study. In Component-Based Software Engineering (pp. 150-165). Springer Berlin Heidelberg.
    - [N+09] Nienaltowski, P., Meyer, B., & Ostroff, J. S. (2009). Contracts for concurrency. Formal Aspects of Computing, 21(4), 305-318.

- This week's **required content**
  - Add design-by-contract to Coordinate interface and class hierarchy
    - Use assert statements and assertion methods for both
      - Preconditions
      - Postconditions
    - Implement `assertClassInvariants` methods for class invariants
- Build, commit, and tag
- Submit this week's summary

# CW #08 Class Preparation Due

- **Class preparation (in recommended order)**

- **Required reading**

- None

- **Suggested reading**

- [A+04] Avižienis, A., Laprie, J. C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. Dependable and Secure Computing, IEEE Transactions on, 1(1), 11-33.
    - [GT07] Grottke, M., & Trivedi, K. S. (2007). Fighting bugs: Remove, retry, replicate, and rejuvenate. Computer, 40(2), 107-109.

# CW #09 Homework Due 1 / 2

- This week's **required content**
  - Add error and exception handling to your classes
- Build, commit, push, and tag
- Submit this week's summary

- Your classes are, at a minimum
  - The Coordinate and related classes
  - Your Photo and related classes
- Review the contracts
  - Associated with your classes
  - And extend them, if necessary
- Determine components boundaries
  - Persistence, domain model, user interface
  - And implement appropriate error handling

# CW #09 Class Preparation Due

- **Class preparation (in recommended order)**

- **Required reading**

- [M82] MacLennan, B. J. (1982). Values and objects in programming languages. ACM SIGPLAN Notices, 17(12), 70-79.

- **Suggested reading**

- [R06] Riehle, D. (2006). Value object. In Proceedings of the 2006 conference on pattern languages of programs (p. 30). ACM.
    - [E04a] Evans, E. (2004). Entities vs. values. Chapter 5 in [E04].

# CW #10 Homework Due

- This week's **required content**
  - Turn the **Coordinate** classes into **value object classes**
    - All **Coordinate** classes should be **immutable** and **shared**
    - Make sure that **Coordinate** objects are still **interchangeable**
- Build, commit, and tag
- Submit this week's summary

# CW #10B Class Preparation Due

- **Class preparation (in recommended order)**

- **Required reading**

- [G+95a] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Introduction. Chapter 1 in [G+95].

- **Suggested reading**

- [RZ96] Riehle, D., & Züllighoven, H. (1996). Understanding and using patterns in software development. TAPOS, 2(1), 3-13.
    - [V98a] Vlissides, J. (1998). Top ten misconceptions. Chapter 1 in [V98].
    - [R03] Riehle, D. (2003). The perfection of informality: Tools, templates, and patterns. Cutter IT Journal, 16(9), 22-26.



# CW #11 Homework Due 1 / 2

- This week's **required content**
  - Document five design pattern instances in Wahlzeit using annotations
  - Explain the purpose of the design patterns in the context of Wahlzeit
- Build, commit, push, and tag
- Submit this week's summary

# CW #11 Homework Due 2 / 2

- Emulate the following example

```
@PatternInstance(  
    patternName = "Abstract Factory"  
    participants = {  
        "AbstractFactory", "ConcreteFactory"  
    }  
)  
public class PhotoFactory { ... }
```

```
@PatternInstance(  
    patternName = "Abstract Factory"  
    participants = {  
        "AbstractProduct",  
        "ConcreteProduct"  
    }  
)  
public class Photo { ... }
```

# CW #11 Class Preparation Due

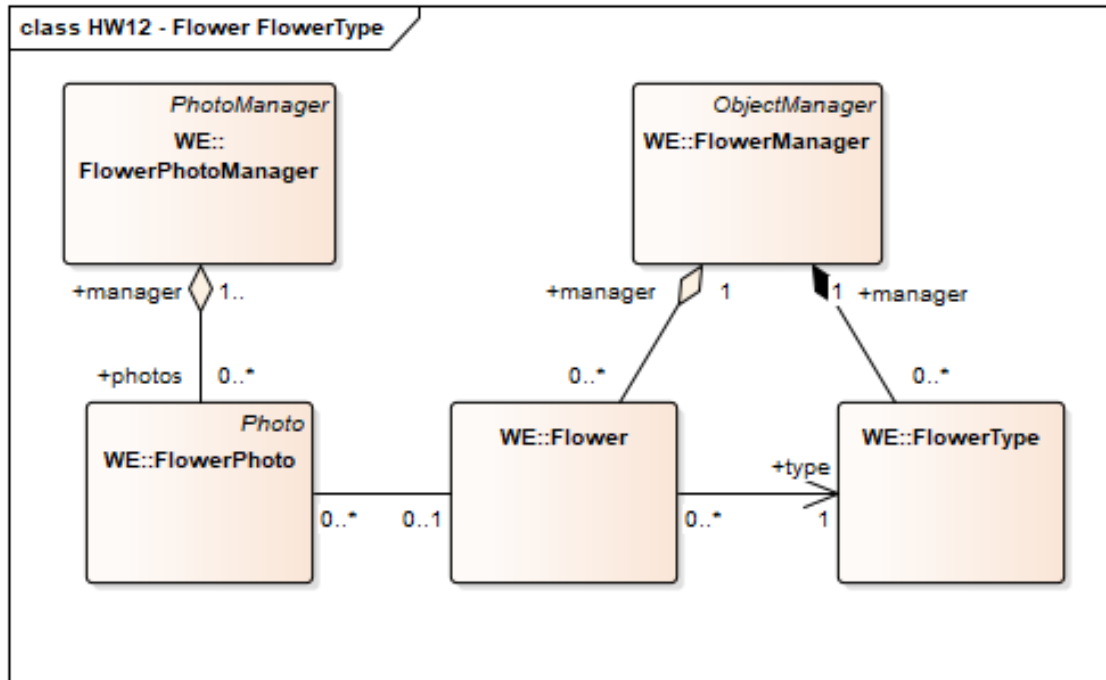
- **Class preparation (in recommended order)**
  - **Required reading**
    - [H95a] Hay, D. C. (1995). Things of the enterprise: Products and product types. Chapter 4 in [H95].
  - **Suggested reading**
    - [JW98] Johnson, R. E., Woolf, B. (1998). Type Object. Chapter 4 in [M+98].

# CW #12 Homework Due 1 / 2

- This week's **required content**
  - Apply the Type Object pattern to your class model
- Build, commit, push, and tag
- Submit this week's summary

# CW #12 Homework Due 2 / 2

- Apply the Type Object pattern to your class model
  - Add your domain class and the corresponding type (object) class
  - Implement an isSubtype() method for your type object class



# CW #12 Class Preparation Due

- **Class preparation (in recommended order)**
  - **Required reading**
    - None
  - **Suggested reading**
    - [G+95b] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Factory Method. Pattern in [G+95].
    - [G+95i] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Abstract Factory. Pattern in [G+95].
    - [BR98] Bäumer, D., Riehle, D. (1998). Product Trader. Chapter 3 in [M+98].
    - [G+95c] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Prototype. Pattern in [G+95].

# CW #13 Homework Due 1 / 2

- This week's **required content**
  - Trace and document the instantiation of both
    - your photo class and
    - your domain class
- Build, commit, push, and tag
- Submit this week's summary

## CW #13 Homework Due 2 / 2

- Trace and document the instantiation process of both
  - your photo class and
  - your domain class
- For each class, document
  - the sequence of method calls that lead to the new object
  - Start with the call to your object manager or, if none, your factory
- For each class, document
  - the object creation solution as a point in the solution space
  - Use the object creation table (i.e. you should provide a six tuple)



# CW #13 Class Preparation Due

- **Class preparation (in recommended order)**

- **Required reading**

- None

- **Suggested reading**

- [R96a] Reenskaug, T. (1996). The Main Ideas. Chapter 1 in [R96].
    - [VN96] VanHilst, M., & Notkin, D. (1996). Using role components in implement collaboration-based designs. ACM SIGPLAN Notices, 31(10), 359-369.

# CW #14 Homework Due 1 / 2

- This week's **required content**
  - Describe the following three collaborations
    - YourObjectPhoto with YourObject
    - YourObject with YourObjectType
    - A collaboration of your choice
- Build, commit, push, and tag
- Submit this week's summary

## CW #14 Homework Due 2 / 2

- Describe the following three collaborations
  - YourObjectPhoto with YourObject
  - YourObject with YourObjectType
  - A collaboration of your choice
- Use the syntax from class, i.e. these keywords
  - `collaboration`, `role`, `binds`, ...

# CW #14 Class Preparation Due

- **Class preparation (in recommended order)**

- **Required reading**

- None

- **Suggested reading**

- [JF88] Johnson, R. E., & Foote, B. (1988). Designing reusable classes. Journal of object-oriented programming, 1(2), 22-35.
    - [WG94] Weinand, A., & Gamma, E. (1994). ET++-a portable, homogenous class library and application framework. Computer Science Research at UBILAB, 66-92.
    - [B+97] Bäumer, D., Gryczan, G., Knoll, R., Lilienthal, C., Riehle, D., & Züllighoven, H. (1997). Framework development for large systems. Communications of the ACM, 40(10), 52-59.

# CW #15 Homework Due

- None

# CW #15 Class Preparation Due

- None

# Thank you! Questions?

**[dirk.riehle@fau.de](mailto:dirk.riehle@fau.de) – <http://osr.cs.fau.de>**

**[dirk@riehle.org](mailto: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
  - Andreas Bauer (2018)