Course Homework

Prof. Dr. Dirk Riehle

Friedrich-Alexander University Erlangen-Nürnberg

ADAP A02

Homework Schedule

- See **Homework** tab at https://oss.cs.fau.de/oss-adap-course

Week Terminology

- This week = current course week (starts with lecture)
- Last week = the course week before this week
- Next week = the course week after this week

Homework Summaries

- Explain your programming homework in two pages
  - Should be appropriately tagged on main branch

- Should have the following header (at max. ½ page):
  - Project name
    - E.g. Flowers
  - Project repository
    - E.g. http://github.com/$STUDNAME/wahlzeit
  - Project's continuous integration information
    - E.g. https://travis-ci.org/$STUDNAME/wahlzeit
  - This week's tag
    - E.g. adap-cw03 on master
  - This week's diff:
    - E.g. https://github.com/$STUDNAME/repository/compare/adap-cw02...$STUDNAME:adap-cw03

- Followed by the explanation of what you did and why (at max. 1.5 page)
  - Please explicitly answer the weekly questions rather than deliver a generic answer

Homework Submission

- Submit this week’s homework as PDF to course management system
  - Submission should not be anonymous
    - Name files lastname-firstname-cwxy.pdf
- Submission closes right before class (submit by then)

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 General

- This week's required content
  - Set-up GitHub account
  - 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, including answers to questions

CW#01 Homework Details

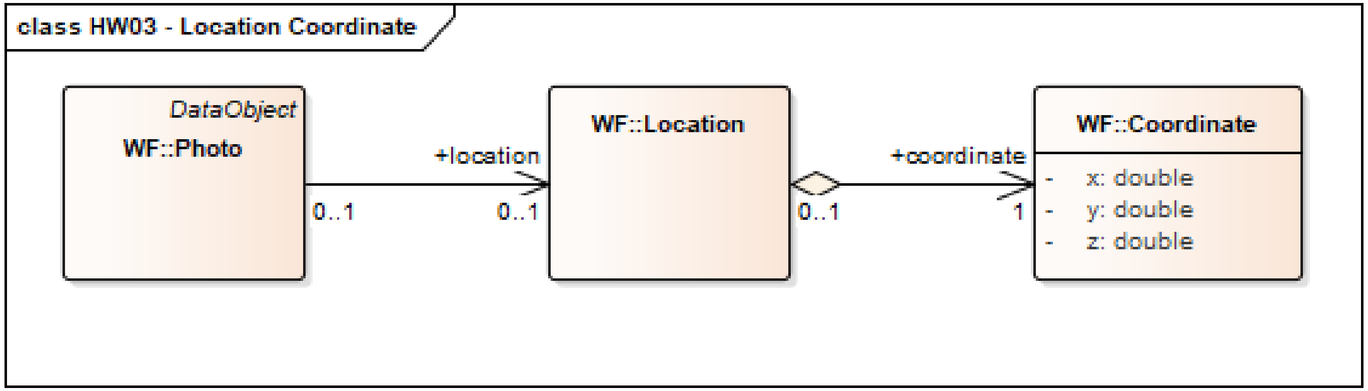
- Compare Dockerfiles
  - There are two Dockerfiles in the root folder of the Wahlzeit project:
    - Dockerfile and simple.Dockerfile
  - Compare the the two Dockerfiles
    - Review the steps taken in both files
    - Explain what is happening
    - Explain the differences between the files
  - Read up about multi-stage Docker builds
  - Explain the advantages of the two-stage build over the simple one in the context of the Wahlzeit
    - Use the provided Dockerfiles as examples!
- Suggested information about multi-stage Docker builds
  - https://docs.docker.com/develop/develop-images/multistage-build/

CW#02 Homework Overview

- This week’s required content
  - Make project decision (type of photo)
  - Add a Location and Coordinate class to Wahlzeit
  - Ensure GitHub Actions are successfully executed (see README.md)
- Build, commit, push, and tag with adap-hw02 to GitHub and DockerHub
- Submit this week’s summary
  - Use https://github.com/$STUDNAME/repository/compare/adap-hw01...$STUDNAME:adap-hw02
  - Keep using this pattern from now on

CW#02 Homework Details

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

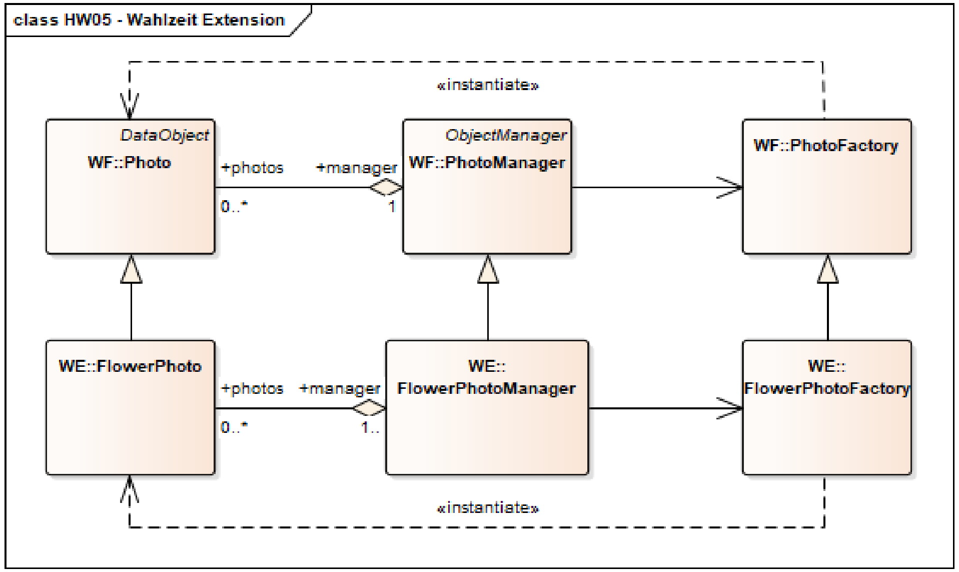


CW#03 Homework Overview

- This week’s required content
  - Extend Wahlzeit with your Photo class; add others if necessary
  - Add test cases for Coordinate, Location, and Photo classes
- Build, commit, push, and tag
- Submit this week’s summary

CW#03 Homework Details 1 / 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#03 Homework Details 2 / 2

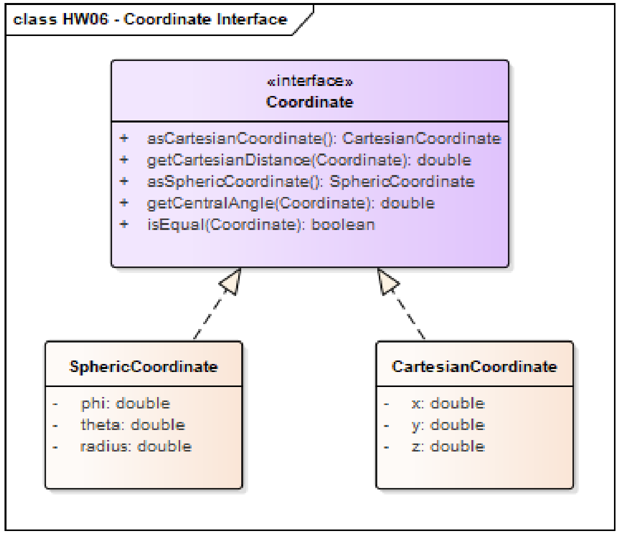
- In your summary, please provide answers to the following questions
  - Why did you extend the Photo class?
  - Why did you not just replace the Photo class?
  - Which tests did you add and why?

CW#04 Homework Overview

- 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#04 Homework Details

- 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 for definition of central angle


class HW06 - Coordinate Interface
«interface»
Coordinate
+ asCartesianCoordinate(): CartesianCoordinate
+ getCartesianDistance(Coordinate): double
+ asSphericCoordinate(): SphericCoordinate
+ getCentralAngle(Coordinate): double
+ isEqual(Coordinate): boolean
SphericCoordinate
- phi: double
- theta: double
- radius: double
CartesianCoordinate
- x: double
- y: double
- z: double

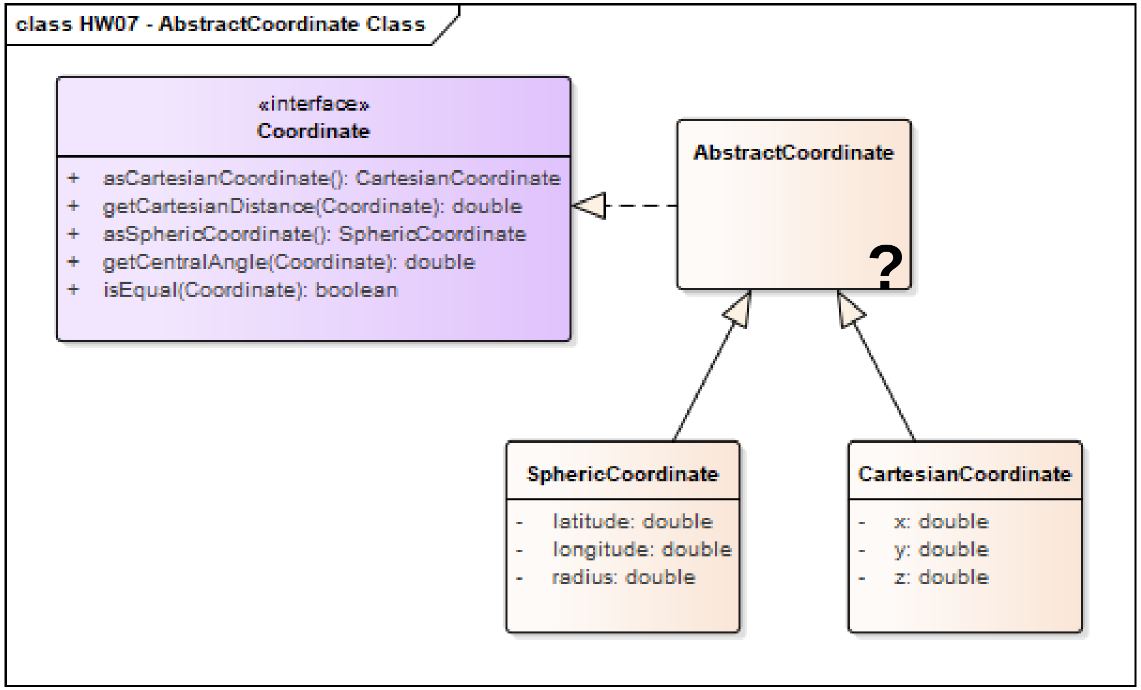
- In your summary, please provide answers to the following questions
  - How did you implement the equality check and why?

CW#05 Homework Overivew

- 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#05 Homework Details

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


class HW07 - AbstractCoordinate Class
«interface»
Coordinate
+ asCartesianCoordinate(): CartesianCoordinate
+ getCartesianDistance(Coordinate): double
+ asSphericCoordinate(): SphericCoordinate
+ getCentralAngle(Coordinate): double
+ isEqual(Coordinate): boolean
AbstractCoordinate
?
SphericCoordinate
- latitude: double
- longitude: double
- radius: double
CartesianCoordinate
- x: double
- y: double
- z: double

- In your summary, please provide answers to the following questions
  - How did you decide which methods go into the abstract superclass or in the implementations?

CW#06 Homework Overview

- 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#06 Homework Details

- In your summary, please provide answers to the following questions
  - How do you test for conditions?
  - Would you rather use Java's `assert` or a dedicated assertion method? Or something else?

CW#07 Homework Overview

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

CW#07 Homework Details

- 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
- In your summary, please provide answers to the following questions
  - What type of exceptions did you use, where, and why?

CW#08 Homework Overview

- 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#08 Homework Details

- In your summary, please provide answers to the following questions
  - What benefits or drawbacks does the Value Object pattern have in this context?
  - How did you handle the mixing of Coordinate objects of different classes and why?

CW#09 Homework Overview

- This week’s required content
  - Document five design pattern instances in Wahlzeit using annotations
- Build, commit, push, and tag
- Submit this week’s summary

CW#09 Homework Details

- Emulate the following example

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

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

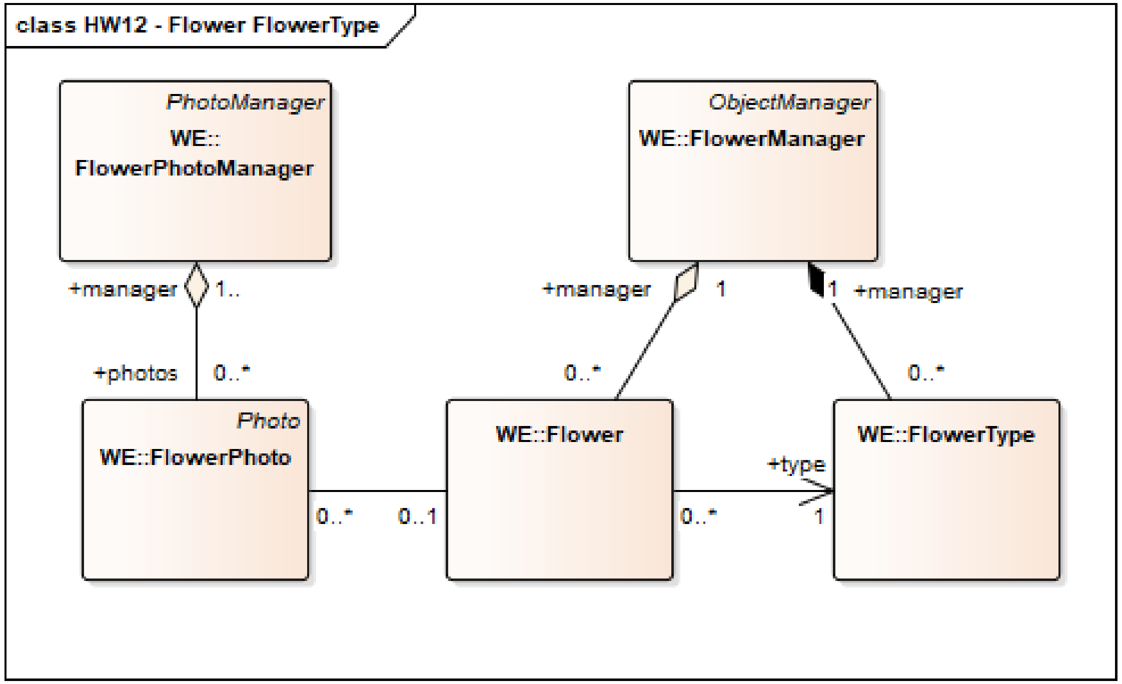
- In your summary, please provide answers to the following questions
  - What is the purpose of the design patterns you documented?
  - Does the use of these design pattern make sense to you? Why?
  - What are the drawbacks of using these design patterns here?

CW#10 Homework Overview

- 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#10 Homework Details

- 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



- In your summary, please provide answers to the following questions
  - How did you deal with type hierarchies?

CW#11 Homework Overview

- 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#11 Homework Details

- 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#12 Homework Overview

- 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#12 Homework Details

- 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`, …
- In your summary, please provide answers to the following questions
  - How to object collaborations relate to design patterns?

Thank you! Questions?

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

dirk@riehle.org – http://dirkriehle.com – @dirkriehle

Credits and License

- Original version
  - © 2012-2020 Dirk Riehle, some rights reserved
  - Licensed under Creative Commons Attribution 4.0 International License
- Contributions
  - Andreas Bauer (2018-2020)
  - Georg Schwarz (2019-2020)