Programming Practices

Prof. Dr. Dirk Riehle

Friedrich-Alexander University Erlangen-Nürnberg

AMOS E02

Licensed under CC BY 4.0 International

General Principles 1 / 4

KISS

(Keep It Simple, Silly)

© 2020 Dirk Riehle - Some Rights Reserved

General Principles 2 / 4

YAGNI

(You Ain’t Gonna Need It)

© 2020 Dirk Riehle - Some Rights Reserved

General Principles 3 / 4

DRY

(Don’t Repeat Yourself)

© 2020 Dirk Riehle - Some Rights Reserved

General Principles 4 / 4

1. **Make it run**
2. **Make it right**
3. **Make it fast**

© 2020 Dirk Riehle - Some Rights Reserved

Definition of Big Ball of Mud [FY97]

A Big Ball of Mud is a haphazardly structured, sprawling, sloppy, duct-tape-and-baling-wire, spaghetti-code jungle. These systems show unmistakable signs of unregulated growth, and repeated, expedient repair. Information is shared promiscuously among distant elements of the system, often to the point where nearly all the important information becomes global or duplicated. The overall structure of the system may never have been well defined. If it was, it may have eroded beyond recognition. Programmers with a shred of architectural sensibility shun these quagmires. Only those who are unconcerned about architecture, and, perhaps, are comfortable with the inertia of the day-to-day chore of patching the holes in these failing dikes, are content to work on such systems.

Benefits of Good (“Well-Factored”) Code

1. **Maintainability**
   (Easier to understand)
2. **Extensibility**
   (Easier to adapt and evolve)
3. **Predictability**
   (Improves planning ability)

© 2020 Dirk Riehle - Some Rights Reserved

Video on Technical Debt [1]

Technical Debt

(Ward Cunningham)

[1] See https://youtu.be/Jp5japiHAs4

© 2020 Dirk Riehle - Some Rights Reserved

Video Lessons

- Technical Debt is a Metaphor
  - Communicates well to manager
  - (Certainly in financial services)
- Taking on debt can speed up development
  - It may be justified to learn faster
  - But you have to pay up later
- If you don’t pay back debt you’ll slow down
  - Paying up means refactoring
  - Still, never write poor code deliberately

© 2020 Dirk Riehle - Some Rights Reserved

Handling Technical Debt

1. **Identify problem**
   (So-called “code smells”)
2. **Identify need to act**
   (Correlate occurrences)
3. **Know how to act**
   (Refactor code)

© 2020 Dirk Riehle - Some Rights Reserved

Code Smell

- According to Fowler [F99]
  - “smells are certain structures in the code that indicate violation of fundamental design principles and negatively impact design quality”
  - also, “a code smell is a surface indication that usually corresponds to a deeper problem in the system”
- Code smells are not bugs

© 2020 Dirk Riehle - Some Rights Reserved

Example Code Smells

- Duplicated Code
- Long Method
- Large Class
- ...

© 2020 Dirk Riehle - Some Rights Reserved

Refactoring (Practice)

- Definition and purpose
  - Is a behavior-preserving transformation of existing source code
  - It is a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior
- More on refactoring
  - Change the structure of code without changing behavior
  - Focus on non-functional features within range of specification
  - Are defined techniques that are typically language specific
  - Are ideally supported by IDEs, for example, the Eclipse JDT
- Defined by Opdyke [O92], popularized by Fowler [F99]

© 2020 Dirk Riehle - Some Rights Reserved

Example Refactorings

- Rename class
- Pull-up field
- Extract method
- …

- More at http://refactoring.com

© 2020 Dirk Riehle - Some Rights Reserved

Example Extract-Method Refactoring 1 / 2

```
public class PhotoManager extends ObjectManager {
     protected Map<PhotoId, Photo> allPhotos = new HashMap<PhotoId, Photo>();

     public void addPhoto(Photo photo) {
          PhotoId id = photo.getId();
          assertIsNewPhoto(id);
          allPhotos.put(id, photo);
          ...
     }

     public Photo getPhotoFromId(PhotoId id) {
          Photo result = doGetPhotoFromId(id);
          if (result == null) {
               ...
               if (result != null) { allPhotos.put(id, result); }
          }
          return result;
     }

     public Set<Photo> findPhotosByOwner(String ownerName) {
          ...
          for (Iterator<Photo> i = r.iterator(); i.hasNext();) {
               Photo photo = i.next();
               allPhotos.put(photo.getId(), photo);
          }
          return r;
     }
     ...
```

Example Extract-Method Refactoring 2 / 2

```
public class PhotoManager extends ObjectManager {

     public void addPhoto(Photo photo) {
          ...
          doAddPhoto(photo);
          ...
     }

     public Photo getPhotoFromId(PhotoId id) {
          ...
          doAddPhoto(photo);
          ...
     }

     public Set<Photo> findPhotosByOwner(String ownerName) {
          ...
          for (Iterator<Photo> i=r.iterator(); i.hasNext(); ) {
               doAddPhoto(i.next());
          }
          ...
     }

     protected void doAddPhoto(Photo myPhoto) {
          allPhotos.put(myPhoto.getId(), myPhoto);
     }

     ...
```

When to Refactor?

The “three strikes” rule

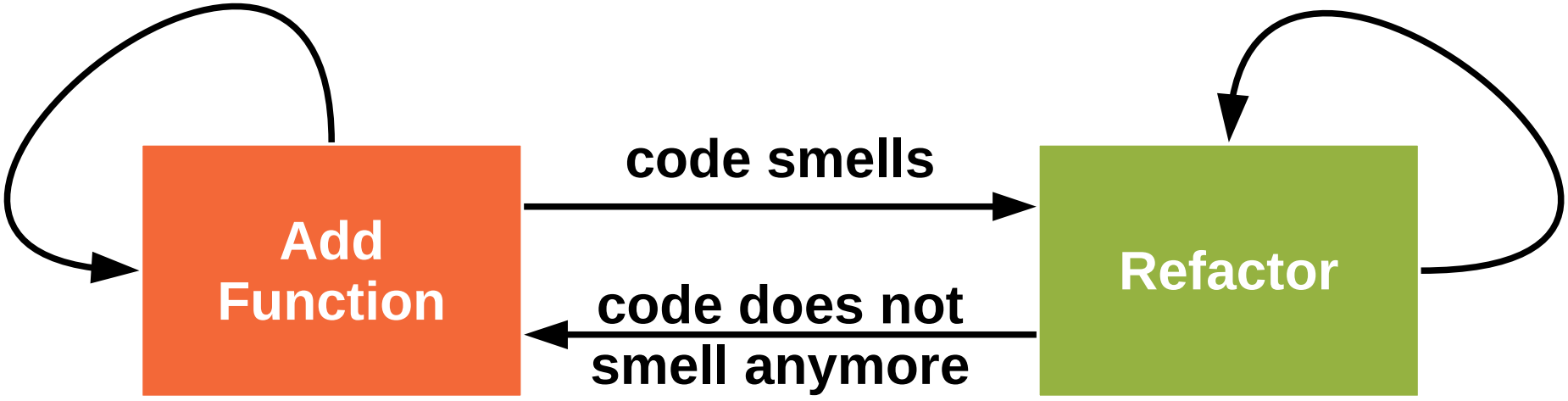
1st time: Just do it

2nd time: Wince at duplication

3rd time: Refactor

© 2020 Dirk Riehle - Some Rights Reserved

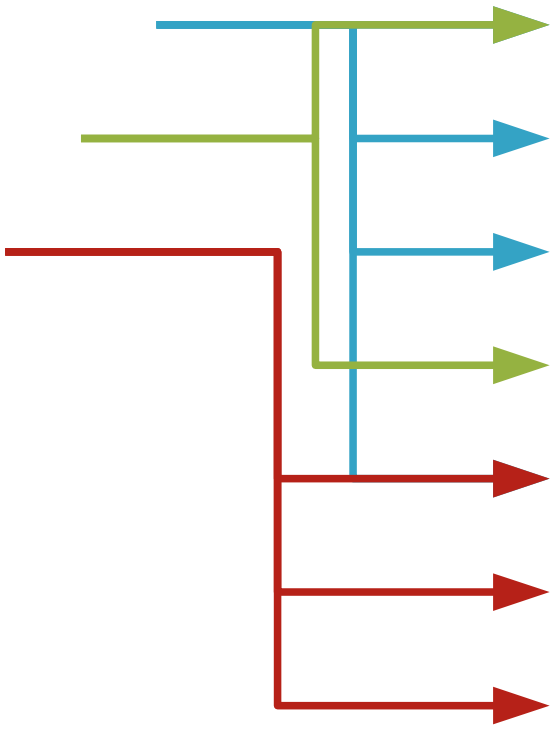
Refactoring Process (Two Hats)



© 2020 Dirk Riehle - Some Rights Reserved

Code Smells and Refactorings

- Duplicated code
- Long method
- Large class
- ...



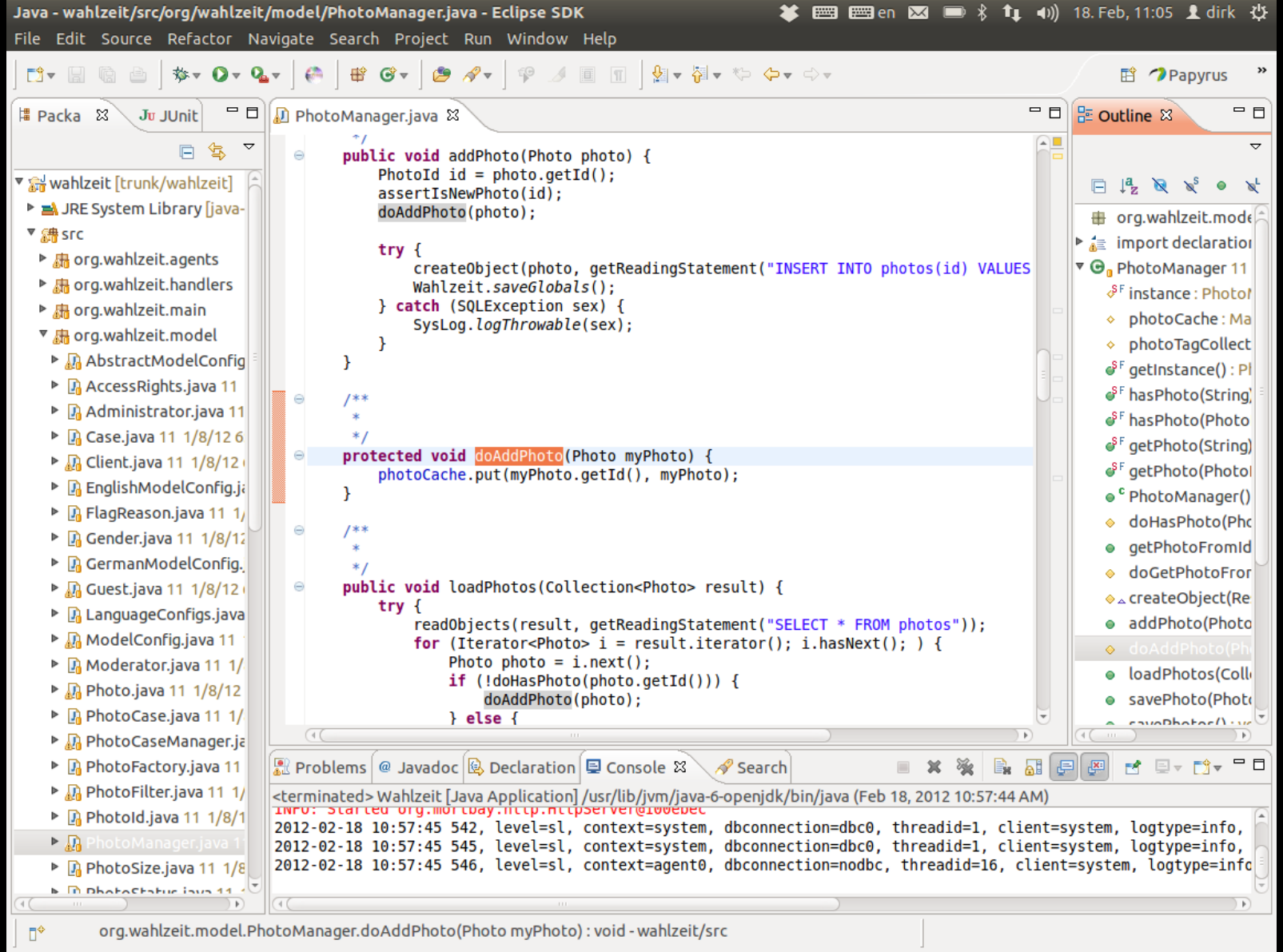
- Extract method
- Pull up field
- Form template method
- Decompose conditional
- Extract class
- Extract subclass
- Extract interface
- ...

© 2020 Dirk Riehle - Some Rights Reserved

Example Refactoring Process

- The refactoring process can become complex
  - It may turn into a series of refactorings
- Example Removal of Switch Statement
  - Extract Method
  - Move Method
  - Replace Type Code …
    - with Subclass or
    - with State / Strategy
  - Replace Conditional ...
    - with Polymorphism

© 2020 Dirk Riehle - Some Rights Reserved


Java - wahlzeit/src/org/wahlzeit/model/PhotoManager.java - Eclipse SDK
18. Feb, 11:05
dirk
File Edit Source Refactor Navigate Search Project Run Window Help
Papyrus
Packa
JUnit
wahlzeit [trunk/wahlzeit]
JRE System Library [java-
src
org.wahlzeit.agents
org.wahlzeit.handlers
org.wahlzeit.main
org.wahlzeit.model
AbstractModelConfig
AccessRights.java 11
Administrator.java 11
Case.java 11 1/8/12 6
Client.java 11 1/8/12
EnglishModelConfig.ja
FlagReason.java 11 1/
Gender.java 11 1/8/12
GermanModelConfig.
Guest.java 11 1/8/12
LanguageConfigs.java
ModelConfig.java 11
Moderator.java 11 1/
Photo.java 11 1/8/12
PhotoCase.java 11 1/
PhotoCaseManager.ja
PhotoFactory.java 11
PhotoFilter.java 11 1/
PhotoId.java 11 1/8/1
PhotoManager.java 1
PhotoSize.java 11 1/8
PhotoManager.java
*/
public void addPhoto(Photo photo) {
PhotoId id = photo.getId();
assertIsNewPhoto(id);
doAddPhoto(photo);
try {
createObject(photo, getReadingStatement("INSERT INTO photos(id) VALUES
Wahlzeit.saveGlobals();
} catch (SQLException sex) {
SysLog.logThrowable(sex);
}
}
/**
*
*/
protected void doAddPhoto(Photo myPhoto) {
photoCache.put(myPhoto.getId(), myPhoto);
}
/**
*
*/
public void loadPhotos(Collection<Photo> result) {
try {
readObjects(result, getReadingStatement("SELECT * FROM photos"));
for (Iterator<Photo> i = result.iterator(); i.hasNext();) {
Photo photo = i.next();
if (!doHasPhoto(photo.getId())) {
doAddPhoto(photo);
} else {
Outline
org.wahlzeit.mode
import declaratio
PhotoManager 11
instance : Photo
photoCache : Ma
photoTagCollect
getInstance() : Pl
hasPhoto(String)
hasPhoto(Photo
getPhoto(String)
getPhoto(Photo
PhotoManager()
doHasPhoto(Phc
getPhotoFromId
doGetPhotoFror
createObject(Re
addPhoto(Photo
doAddPhoto(Ph
loadPhotos(Coll
savePhoto(Phot
Problems
Javadoc
Declaration
Console
Search
<terminated> Wahlzeit [Java Application] /usr/lib/jvm/java-6-openjdk/bin/java (Feb 18, 2012 10:57:44 AM)
2012-02-18 10:57:45 542, level=sl, context=system, dbconnection=dbc0, threadid=1, client=system, logtype=info,
2012-02-18 10:57:45 545, level=sl, context=system, dbconnection=dbc0, threadid=1, client=system, logtype=info,
2012-02-18 10:57:45 546, level=sl, context=agent0, dbconnection=nodbc, threadid=16, client=system, logtype=info
org.wahlzeit.model.PhotoManager.doAddPhoto(Photo myPhoto) : void - wahlzeit/src

Quiz on Refactoring

- Your code smells. All signs for refactoring are given. Under which circumstances should you not start a refactoring?

© 2020 Dirk Riehle - Some Rights Reserved

Review / Summary of Session

- Agile principles of programming
  - KISS, YAGNI, DRY
  - Make it run, right, fast
- Technical debt and refactoring
  - Principles, two hats
  - Example refactorings and tools

© 2020 Dirk Riehle - Some Rights Reserved

Thank you! Questions?

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

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

Credits and License

- Original version
  - © 2020 Dirk Riehle, some rights reserved
  - Licensed under Creative Commons Attribution 4.0 International License
- Contributions
  - None yet