Công Cụ & Phương Pháp Thiết Kế - Quản Lý (Phần Mềm)

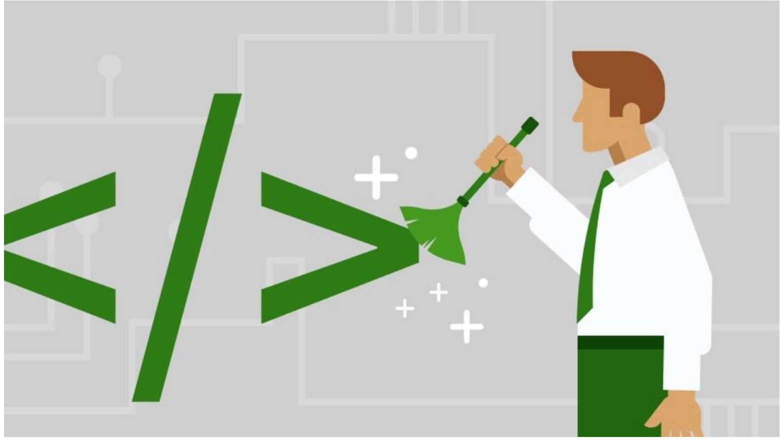
TRAN KIM SANH
Instructor of DTU

Email: trankimsanh@dtu.edu.vn
Tel: 0987 409 464

Refactoring

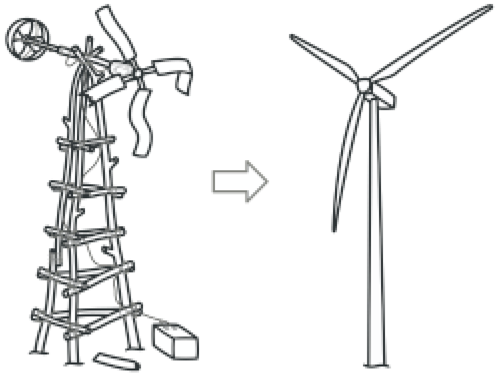
Director’s Overview

- What is Refactoring?
- Why Refactor?
- Reasons Not to Refactor?
- How to Refactor
- Refactoring in eclipse



What is Refactoring?

- “Refactoring" source code means improving it without changing what it does
- Refactoring does NOT:
  - Fix defects
  - Add new functionality



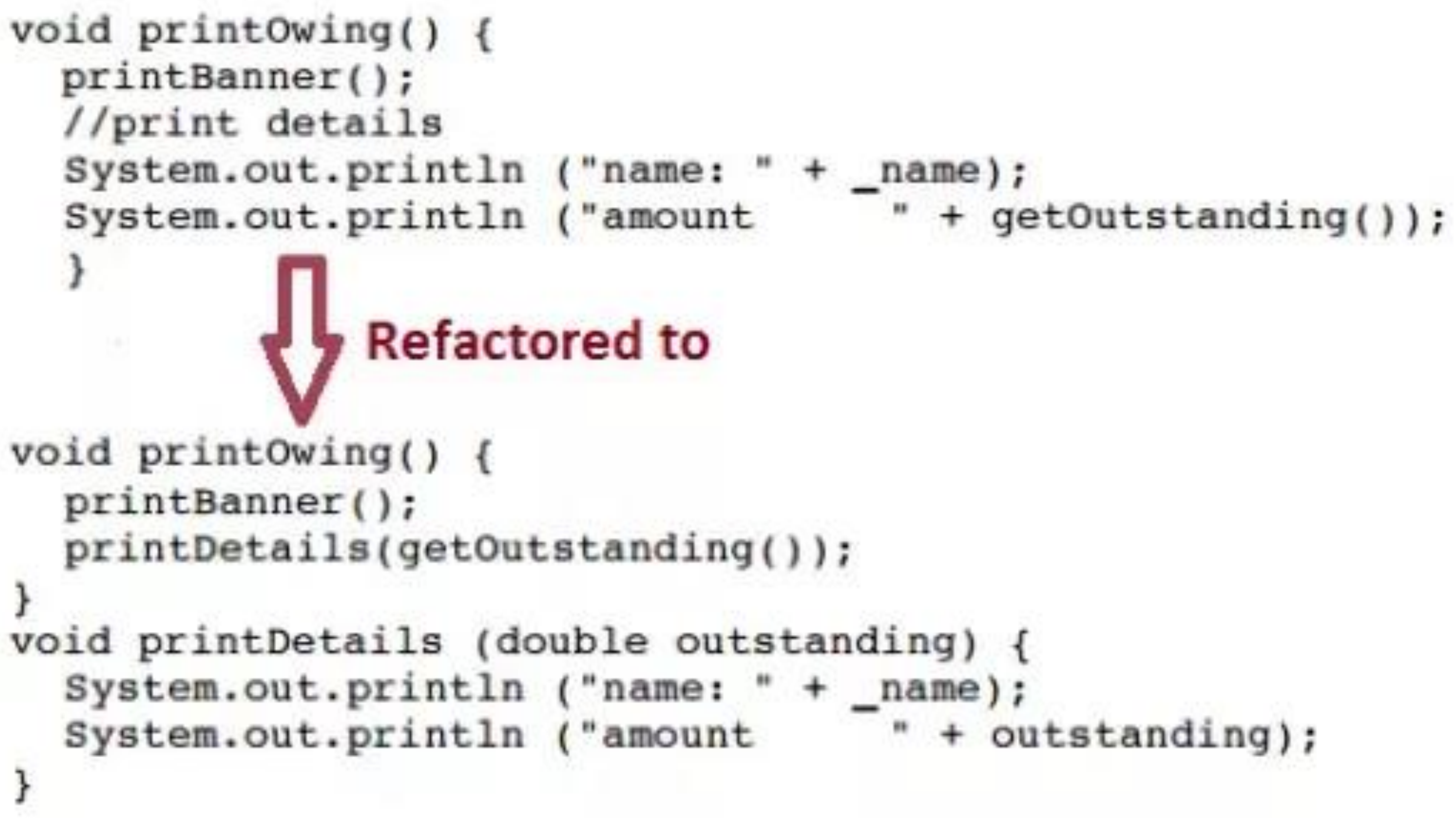
- The goal of refactoring is to:
  - Improve the understandability of the code
  - Improve the structure of the code
  - Remove unnecessary code

What is Refactoring?

- Refactoring is a disciplined technique
  - Each refactor should be small
    - ✓ So it is less likely to go wrong
  - The system is kept fully working after each small refactoring
    - ✓ Reducing the chances that a system can get seriously broken during the restructuring
    - ✓ Increasing the need for automated unit test
- Refactoring can produce a significant benefit over time

Martin Fowler, http://www.refactoring.com/

Example



```
void printOwing() {
  printBanner();
  //print details
  System.out.println ("name: " + _name);
  System.out.println ("amount      " + getOutstanding());
  }
```

Refactored to

```
void printOwing() {
  printBanner();
  printDetails(getOutstanding());
}
void printDetails (double outstanding) {
  System.out.println ("name: " + _name);
  System.out.println ("amount      " + outstanding);
}
```

Figure 1: Example of Refactoring

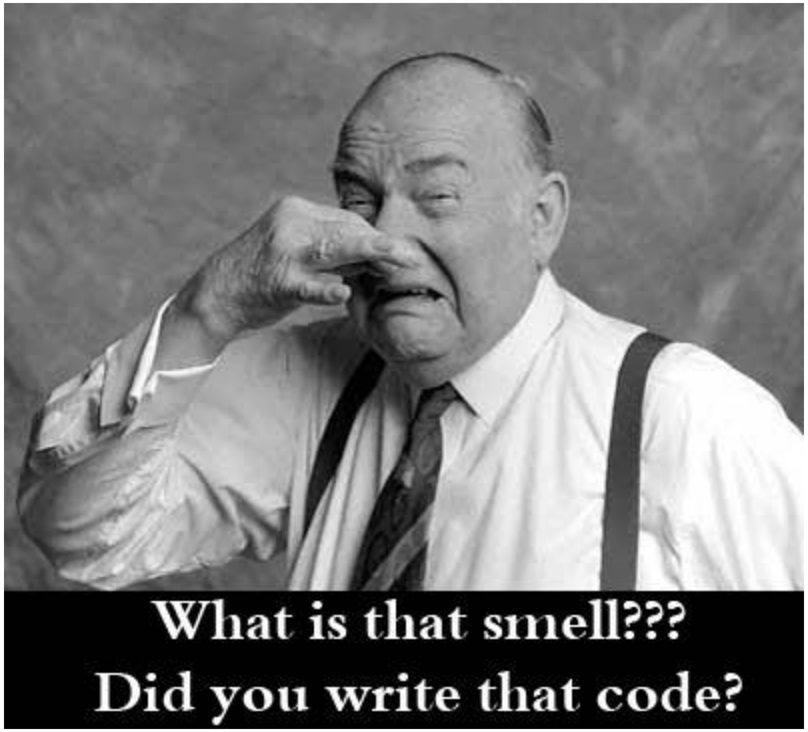
Referenced from Prof.Redley of CMU

Why Refactor?

- Because software evolves over time
  - During development
    - ✓ Original developers involved
  - During maintenance
    - ✓ Different developers likely to be involved
    - ✓ Original intent of developers has been forgotten
- Because of “code smells”
  - Smells are heuristics that can indicate when and what to refactor
  - Smells are indicators, not causes

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

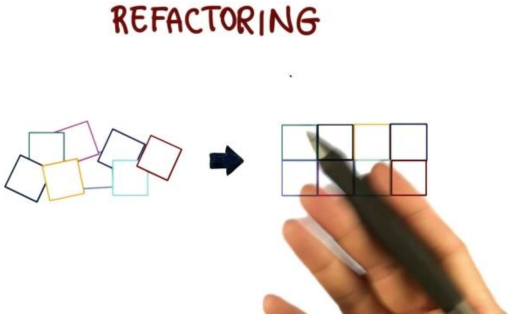
Bad smell of code

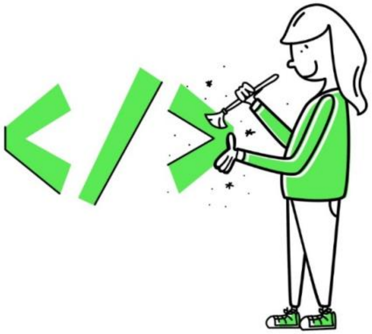


Referenced from Prof.Redley of CMU

Is Your Software's Evolution ...

- A planned-for opportunity that is improving the internal quality of the software?
- A haphazard activity that is continually degrading the product's quality?





Is Quality An Accident?

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

Some Reasons to Refactor - 1

- Duplicate code
  - Must make changes in multiple places
- Routine that is too long
  - Routines should do one thing well
- Loops are too long or too deeply nested
  - Convert loop content into routines?
- Poor cohesion
  - Methods that implement a single function are described as having high cohesion

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

2
n
d

Some Reasons to Refactor - 2

- Class interface with an inconsistent level of abstraction
  - May want to recapture interface integrity
- Parameter list with too many parameters
  - Well-factored programs tend to have many small, well-defined routines that don’t require large parameter lists
- Class changes are compartmentalized
  - A class has too many responsibilities

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

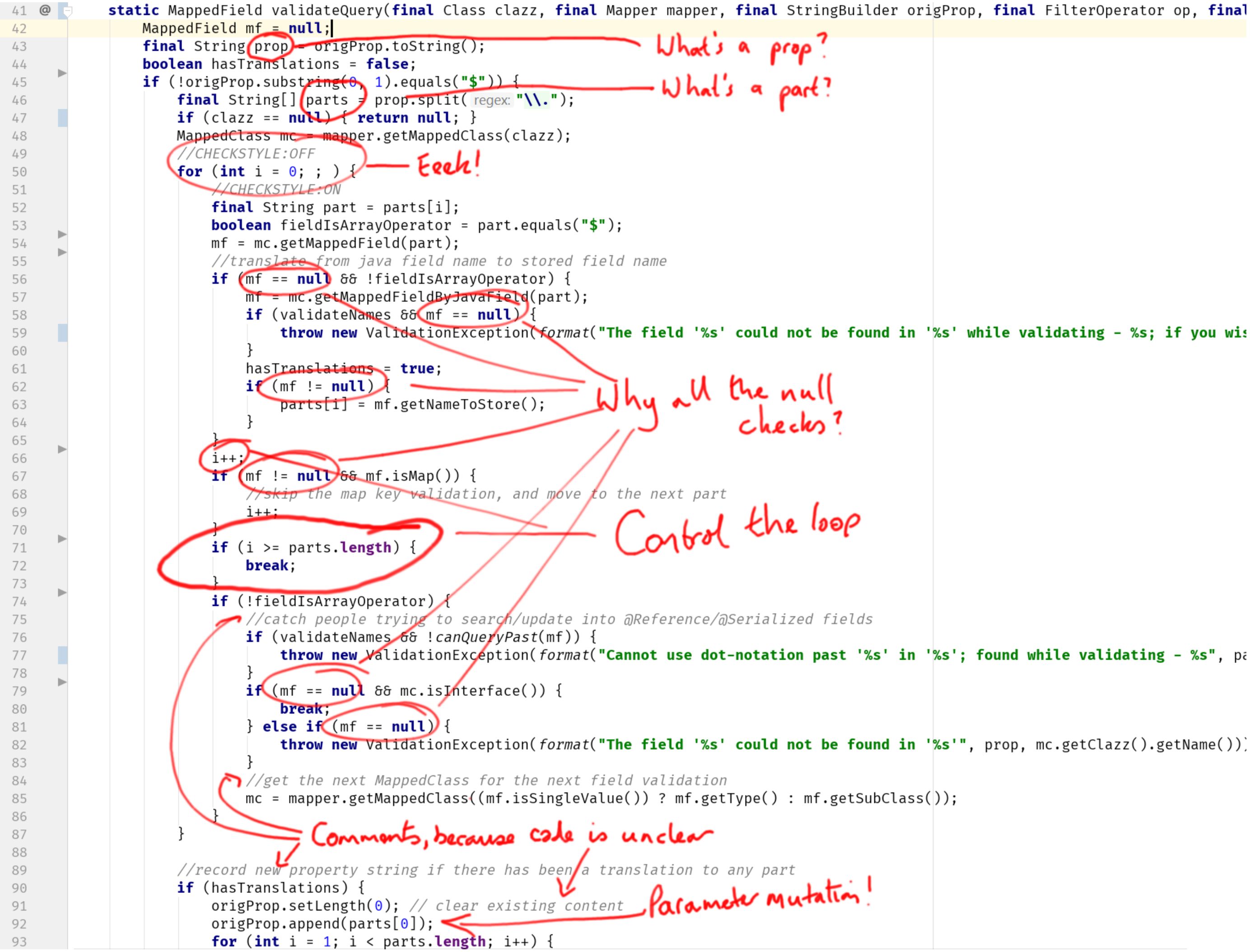
2
n
d

Some Reasons to Refactor - 3

- Parallel modifications to multiple classes
  - Should classes be rearranged so that changes affect only one class
- Inheritance hierarchies have to be modified in parallel
  - Making a subclass of one class every time you make a subclass of another class is another form of parallel modification
- Related data items that are used together are not organized into classes

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

Code smell heuristics



```
static MappedField validateQuery(final Class clazz, final Mapper mapper, final StringBuilder origProp, final FilterOperator op, final
    MappedField mf = null;
    final String prop = origProp.toString();
    boolean hasTranslations = false;
    if (!origProp.substring(0, 1).equals("$")) {
        final String[] parts = prop.split( regex: "\\.");
        if (clazz == null) { return null; }
        MappedClass mc = mapper.getMappedClass(clazz);
        //CHECKSTYLE:OFF
        for (int i = 0; ; ) {
            //CHECKSTYLE:ON
            final String part = parts[i];
            boolean fieldIsArrayOperator = part.equals("$");
            mf = mc.getMappedField(part);
            //translate from java field name to stored field name
            if (mf == null && !fieldIsArrayOperator) {
                mf = mc.getMappedFieldByJavaField(part);
                if (validateNames && mf == null) {
                    throw new ValidationException(format("The field '%s' could not be found in '%s' while validating - %s; if you wis
                }
                hasTranslations = true;
                if (mf != null) {
                    parts[i] = mf.getNameToStore();
                }
            }
            i++;
            if (mf != null && mf.isMap()) {
                //skip the map key validation, and move to the next part
                i++;
            }
            if (i >= parts.length) {
                break;
            }
            if (!fieldIsArrayOperator) {
                //catch people trying to search/update into @Reference/@Serialized fields
                if (validateNames && !canQueryPast(mf)) {
                    throw new ValidationException(format("Cannot use dot-notation past '%s' in '%s'; found while validating - %s", pa
                }
                if (mf == null && mc.isInterface()) {
                    break;
                } else if (mf == null) {
                    throw new ValidationException(format("The field '%s' could not be found in '%s'", prop, mc.getClazz().getName())
                }
                //get the next MappedClass for the next field validation
                mc = mapper.getMappedClass((mf.isSingleValue()) ? mf.getType() : mf.getSubClass());
            }
        }

        //record new property string if there has been a translation to any part
        if (hasTranslations) {
            origProp.setLength(0); // clear existing content
            origProp.append(parts[0]);
            for (int i = 1; i < parts.length; i++) {
```

What's a prop?

What's a part?

Eeek!

Why all the null checks?

Control the loop

Comments, because code is unclear

Parameter mutation!

Referenced from Prof.Redley of CMU

Ex: Data Level Refactoring

- Replace a magic number with a named constant
  - Use a named constant (e.g. “Pi”) instead of a literal (e.g. “3.14”)
- Rename a variable with a clearer or more informative name
  - Replace “name” with “accountname”

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

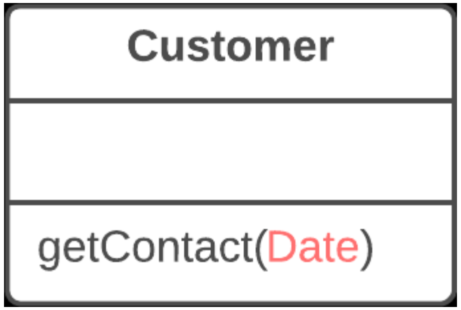
Ex: Statement Level Refactoring

- Consolidate fragments that are duplicated within different parts of a conditional
    - If same lines of code are repeated in a conditional
- Replace conditionals with polymorphism (especially with repeated case statements)
- Create and use null objects instead of testing for null values
    - Move null checking code away from the client and into the class

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

Ex: Routine Level Refactoring

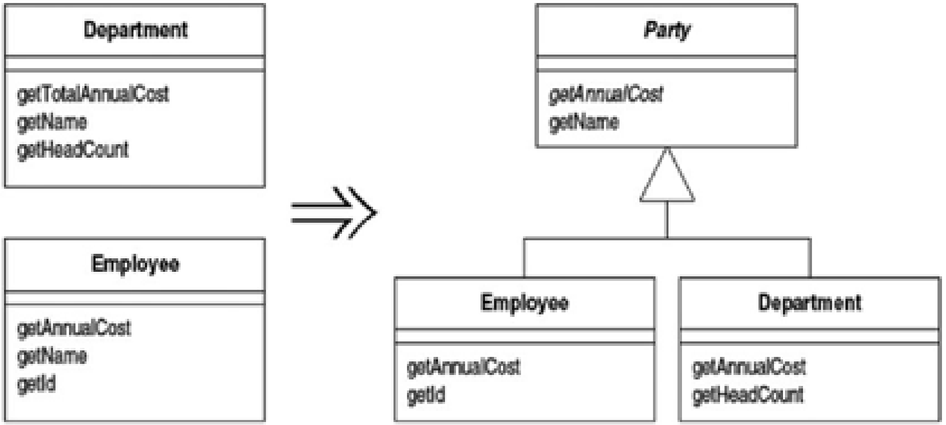
- Extract a routine
  - Remove inline code from one routine, and turn it into its own routine
- Convert a long routine to a class
  - If a routine is too long then maybe it should be its own class
- Substitute a simple algorithm for a complex algorithm
  - Simplify
- Remove a parameter
  - If the parameter is no longer used



Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

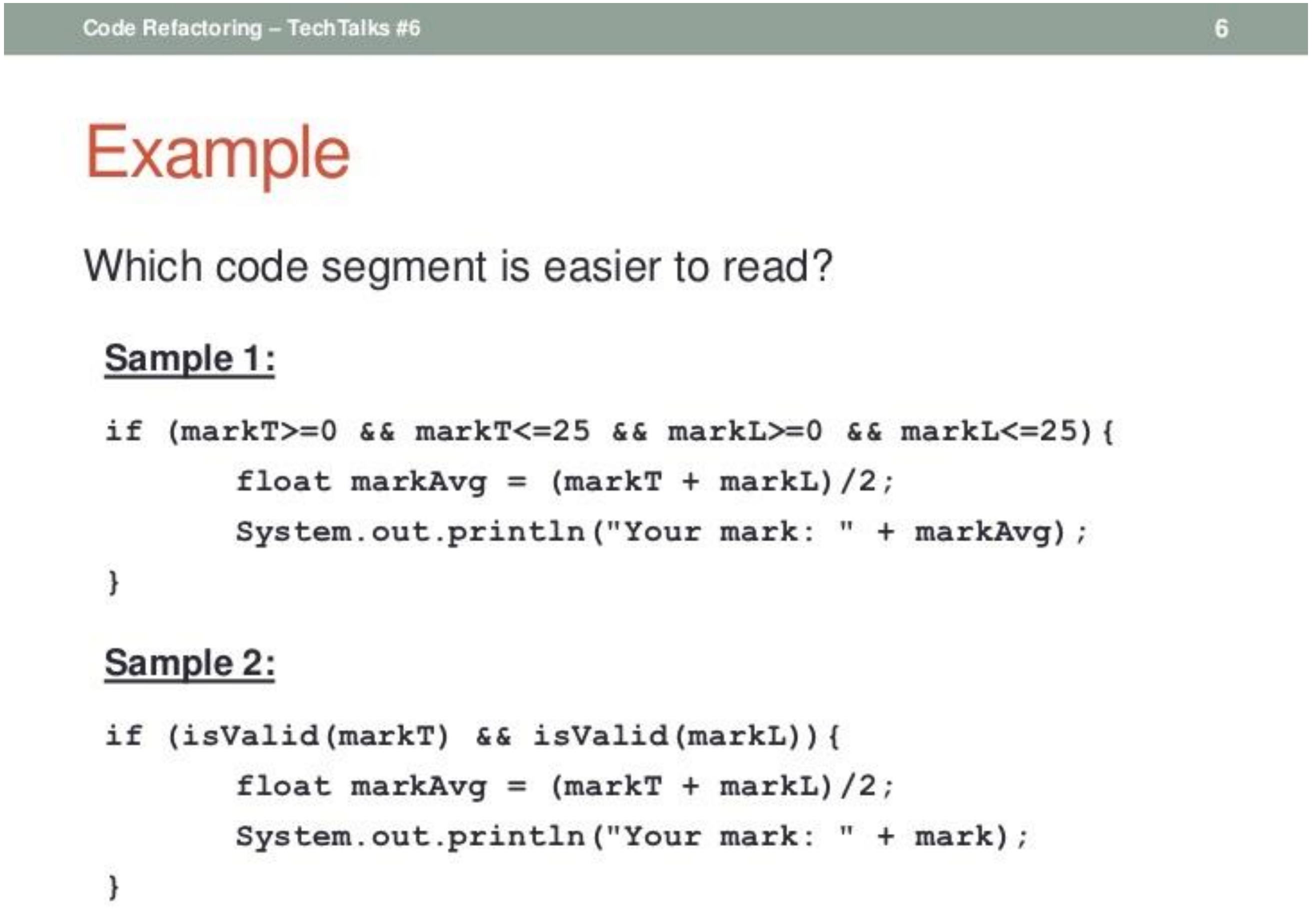
Ex: Class Implementation Refactor

- Extract specialized code into a subclass
  - If a class has code that's used by only a subset of its instances, move that specialized code into its own subclass
- Combine similar code into a superclass
  - If two subclasses have similar code, combine that code and move it into the superclass



Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

Example



Example

Which code segment is easier to read?

Sample 1:

```
if (markT>=0 && markT<=25 && markL>=0 && markL<=25){
        float markAvg = (markT + markL)/2;
        System.out.println("Your mark: " + markAvg);
}
```

Sample 2:

```
if (isValid(markT) && isValid(markL)){
        float markAvg = (markT + markL)/2;
        System.out.println("Your mark: " + mark);
}
```

Referenced from Prof.Redley of CMU

Ex: Class Interface Refactoring

- Convert one class to two
  - If a class has more than one distinct area of resonsibility
- Eliminate a class
  - If the class isn’t doing much
- Encapsulate an exposed member variable
  - Change the data to private and expose the data’s value through a routine instead

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

Ex: System Level Refactoring

- Duplicate data you can't control
  - If you have multiple sources that must access data, then move the data to its own class and have all sources treat that class as the definitive source of the data

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

Reasons NOT to Refactor

- Refactoring is NOT defect fixing, adding functionality or modifying the design
  - Do these types of maintenance efforts separately
- Sometimes code is so bad it needs to be rewritten

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

Reasons NOT to Refactor

```
.author div.hfeed div.archive-meta p{
.single div.navigation{margin:4em 0 0;}
footer{color:#9aa1ae;font-size:0.9em;padding:0em 0 0;}
#footer{color:#9aa1ae;font-size:0.9em;padding:6em 0 0;} span#theme-link span.addition
#globalnav,div.access,div#footer span#theme-link span.addition
#header a{text-decoration:none;}
#header div#blog-description,div#header div.archive-descript
#header h1#blog-title{border-bottom:0.1em solid #ccd0d5;font
#wrapper .important{background:#f5f5dc url(images/important
#wrapper div.entry-content .download{background:url(images/
#wrapper div.entry-content .html{background:url(images/file
#wrapper div.entry-content .html,div#wrapper div.entry-con
#wrapper div.entry-content .pdf{background:url(images/file
#wrapper div.entry-content .zip{background:url(images/file
div.comments h3{color:#4d5663;font-size:1.4em;font-weight:40
div.comments h3.comment-header span.comment-count{color:#
div.comments ol#comments{margin:0 0 0 2
div.comments ol#comments li{margin:1em
div.comments ol#comments li img.photo
div.comments ol#pingbacks{list-style:n
div.comments ol#pingbacks li{border:1
```

Referenced from Prof.Redley of CMU

How to Refactor - 1

- Make sure your can get back to where you started
  - SCM system
  - Backups
- Keep refactoring as small as you can
- Do one refactoring at a time
- Make a list of refactoring steps you intend to take
- Log additional refactoring ideas/needs that you encounter

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

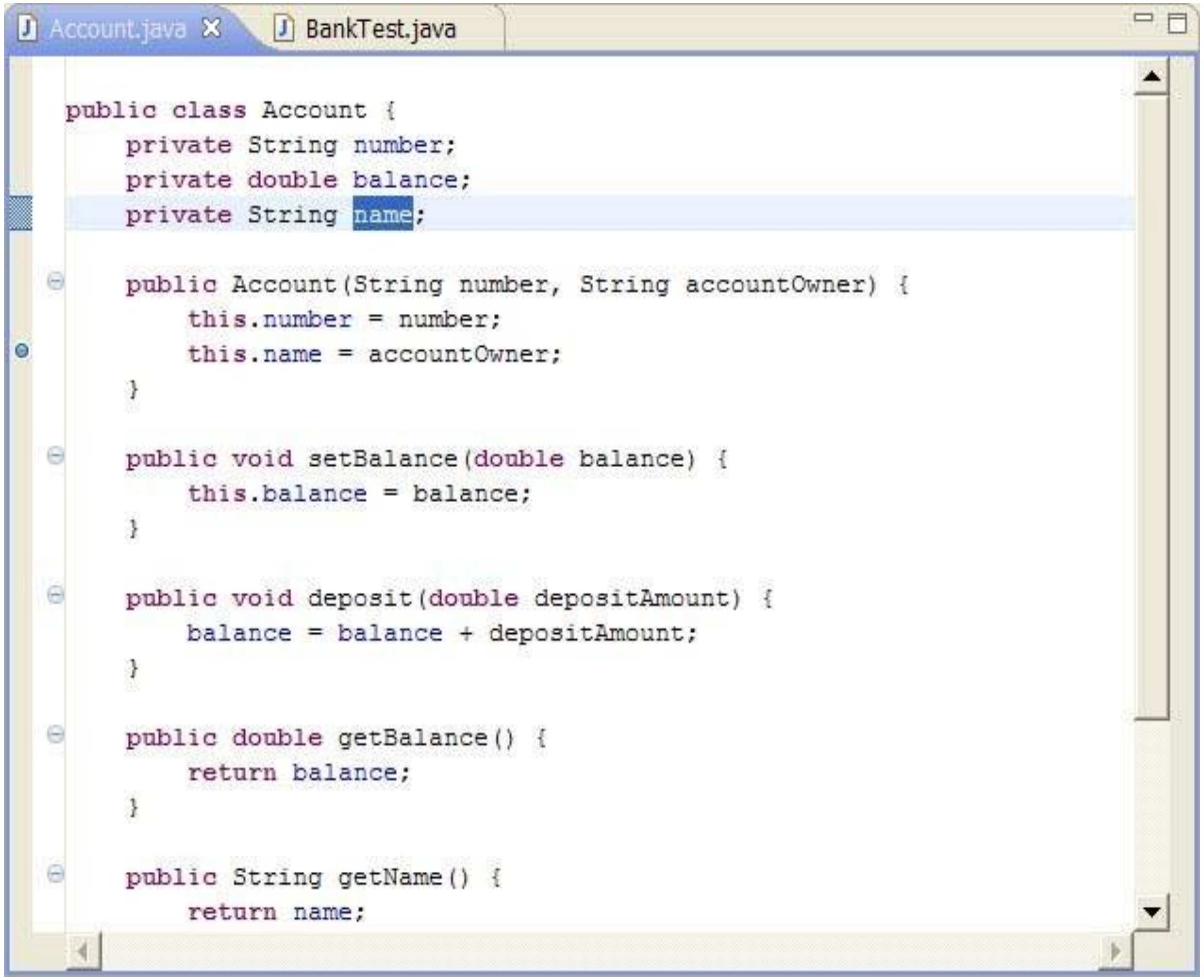
How to Refactor - 2

- Make frequent checkpoints
- Execute existing unit tests
- Add new unit tests
- Recognize that different refactoring efforts include different levels of risk
  - Err on the side of caution
  - Peer review your refactoring changes

Steve McConnell, Code Complete, 2nd Edition. Microsoft Press 2004

Refactoring in eclipse

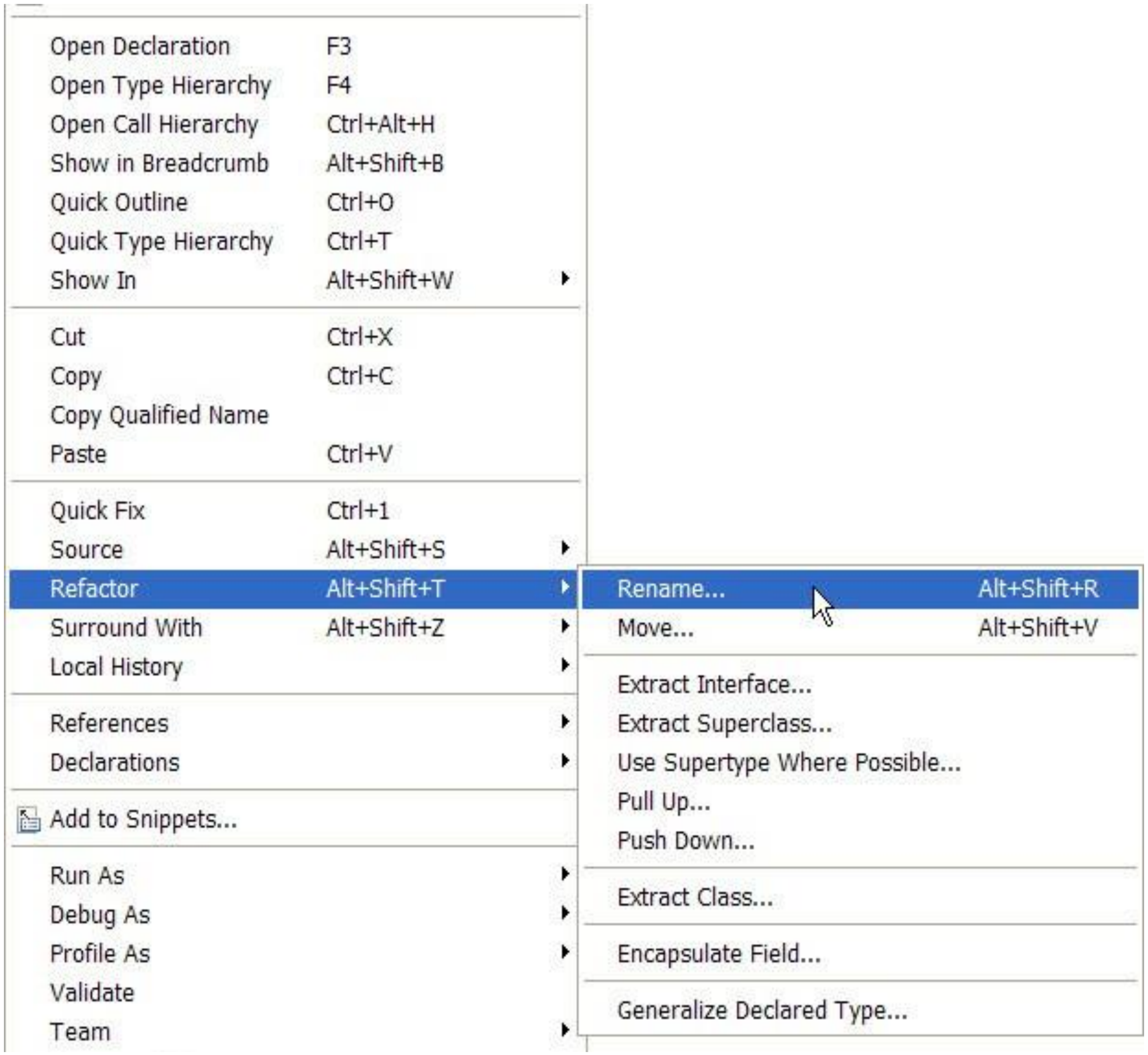
- We have a variable (name) that we want to change to something more meaningful (accountownername)



```java
public class Account {
    private String number;
    private double balance;
    private String name;

    public Account(String number, String accountOwner) {
        this.number = number;
        this.name = accountOwner;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public void deposit(double depositAmount) {
        balance = balance + depositAmount;
    }

    public double getBalance() {
        return balance;
    }

    public String getName() {
        return name;
```

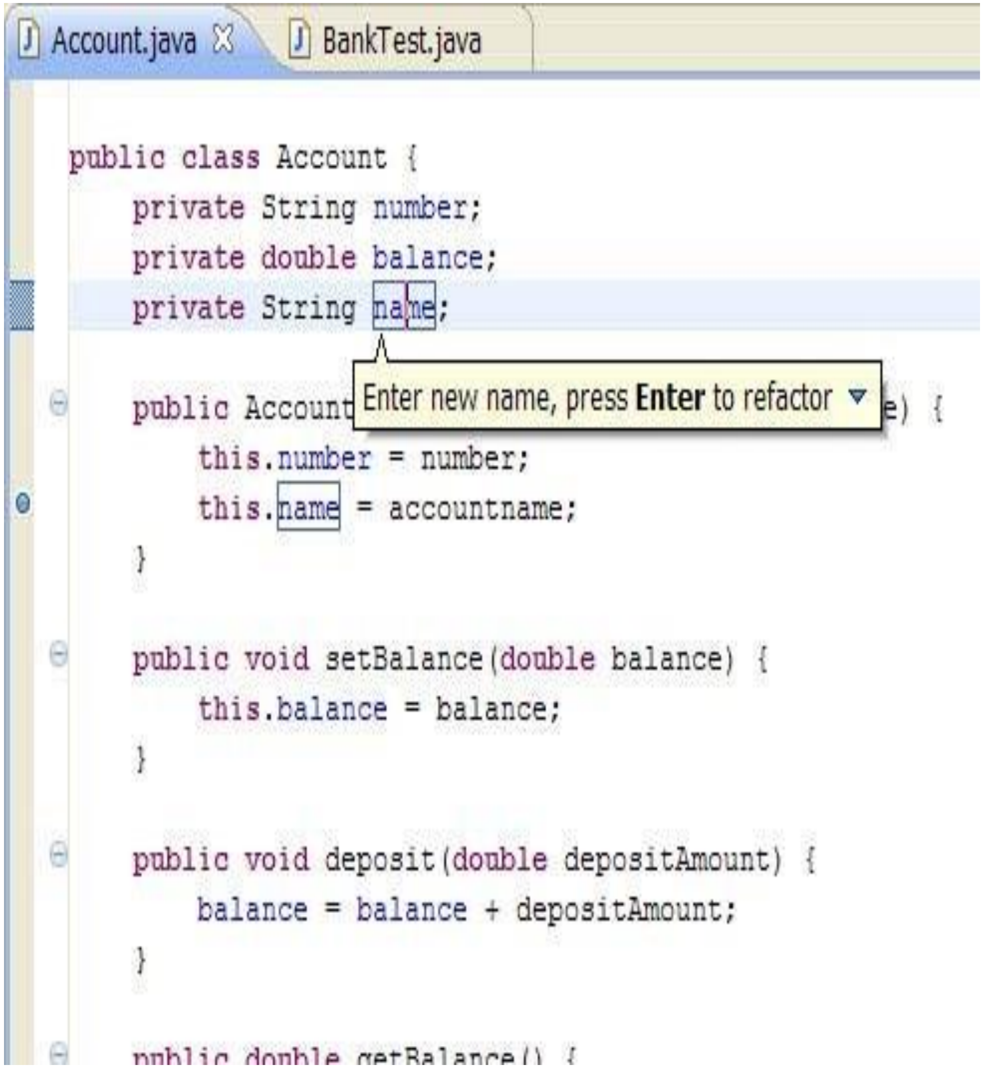
Refactoring Menu in eclipse

- Right click on the variable, select Refactor and then Rename
- Note the other refactoring tools available



Refactoring Menu in eclipse

- Enter the new name for the variable, then press Enter

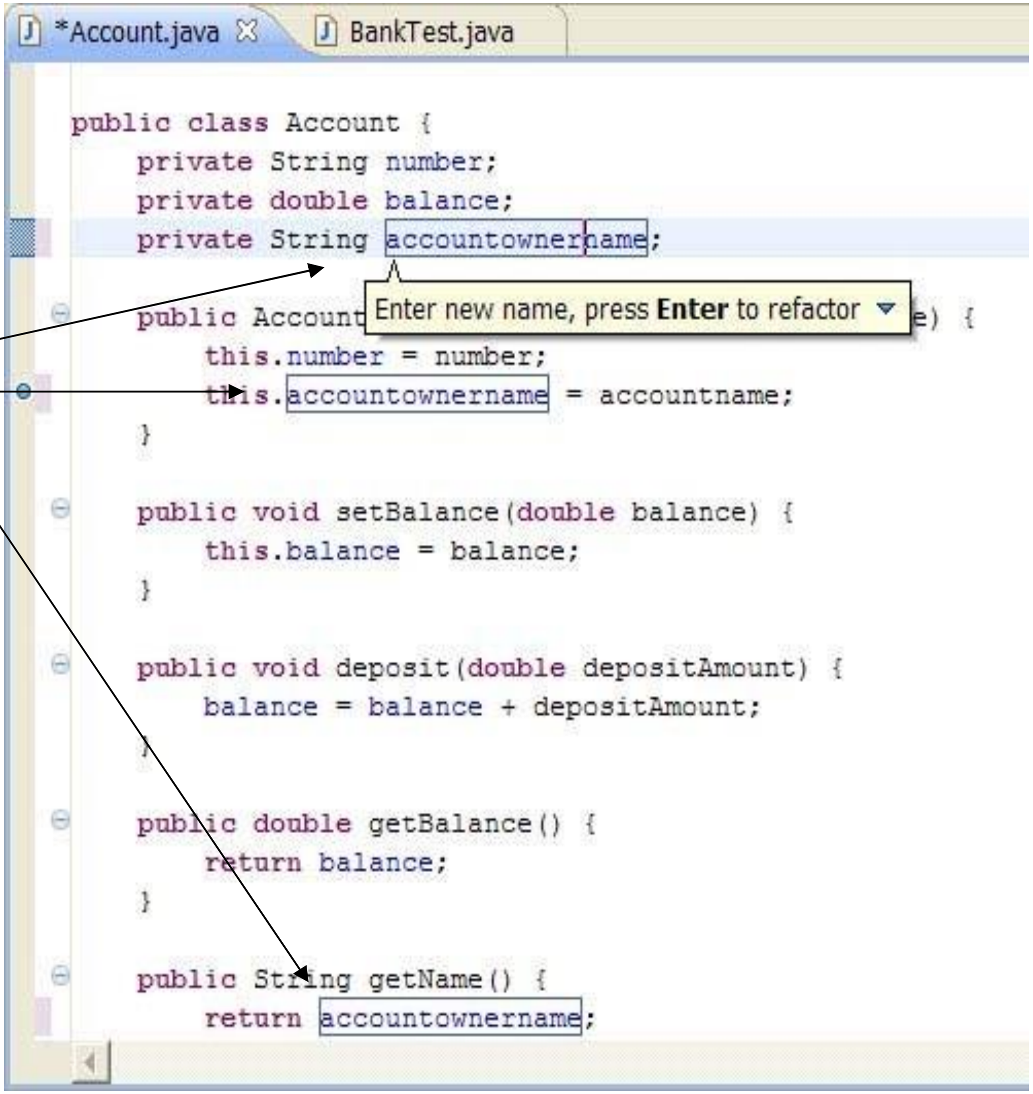


```
public class Account {
    private String number;
    private double balance;
    private String name;

    public Account                                        e) {
        this.number = number;
        this.name = accountname;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public void deposit(double depositAmount) {
        balance = balance + depositAmount;
    }

    public double getBalance() {
```

Enter new name, press **Enter** to refactor

Refactoring Menu in eclipse

- Note that eclipse has found everywhere that the variable is used and changed it there for me



```
public class Account {
    private String number;
    private double balance;
    private String accountownername;

    public Account                                        e) {
        this.number = number;
        this.accountownername = accountname;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public void deposit(double depositAmount) {
        balance = balance + depositAmount;
    }

    public double getBalance() {
        return balance;
    }

    public String getName() {
        return accountownername;
```

Enter new name, press **Enter** to refactor

Summary

- Refactoring does NOT change the software’s functionality
- Refactoring is just good programming
- Be careful and safe when you refactor
- The refactoring lists in this presentation are not complete. For complete lists:
    - Steve McConnell, Code Complete, 2nd edition. Chapter 24 (cc2e.com)
    - Martin Fowler, Refactoring: Improving the Design of Existing Code

Group discussion?

- Discuss with all team member about the refactoring function on eclipse (10 minutes)

Video link

- https://www.youtube.com/watch?v=dIj1W8RKge8

References

- McConnell, Steve Code Complete. Microsoft Press, 2004. pages 563 – 585
- Martin Fowler, Refactoring: Improving the Design of Existing Code