TRASYS

WE GET IT DONE

Java Package by Feature

Java Day 2014, Athens Greece

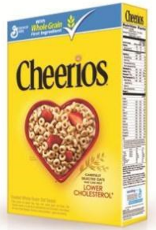
Kostis Kapelonis (Trasys)

An imaginary story

…for a supermarket

I want to buy some toothpaste

Supermarket is organized by color (!!!)



TRASY
WE GET IT DONE

I would have to ask

-Where do I find toothpaste?

-“It depends on the colour”

I would have to ask

-Why by colour?

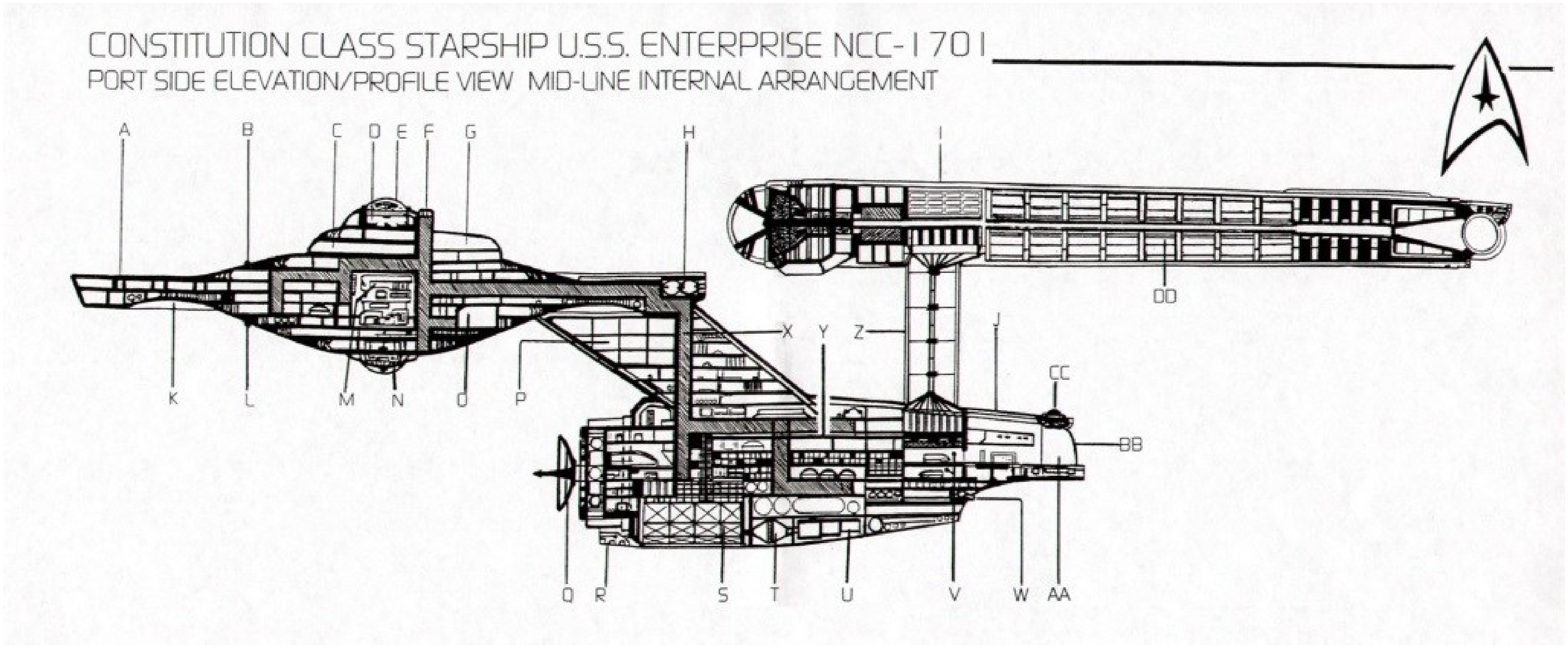
-Because it is very easy for us to load the shelves

WTF?

I would then go to a “normal” supermarket

…where products are grouped by usage

The context

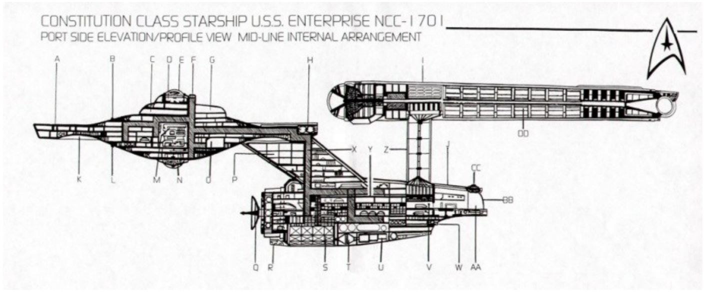

CONSTITUTION CLASS STARSHIP U.S.S. ENTERPRISE NCC-1701
PORT SIDE ELEVATION/PROFILE VIEW MID-LINE INTERNAL ARRANGEMENT
A
B
C
D
E
F
G
H
I
DD
J
X
Y
Z
CC
BB
K
L
M
N
O
P
Q
R
S
T
U
V
W
AA

Java Enterprise Applications

TRASYS
WE GET IT DONE

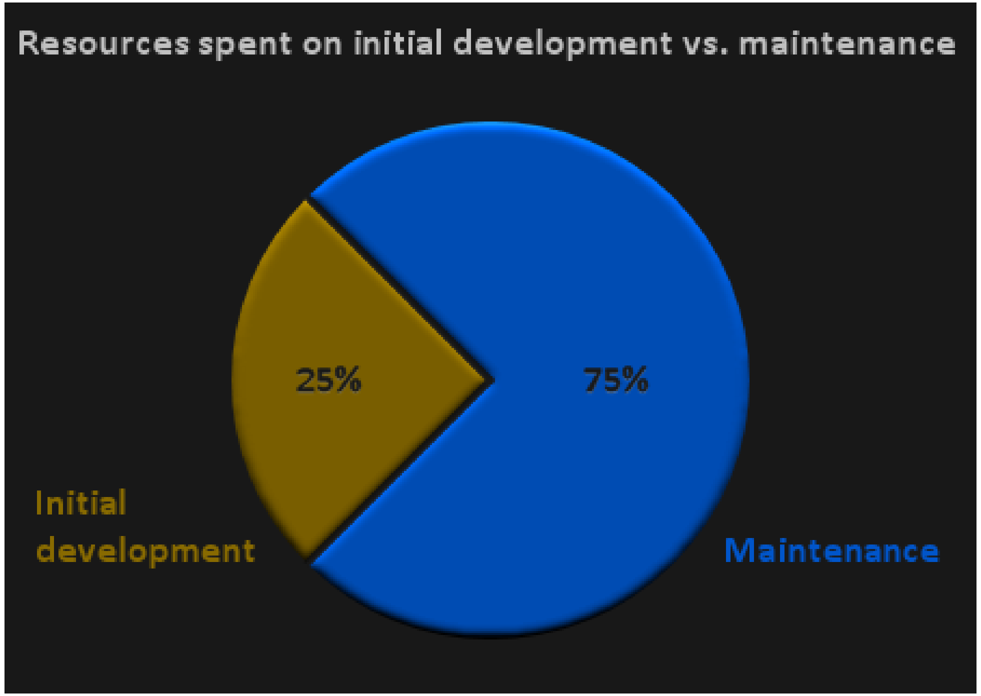
Java Enterprise applications

- Big codebase (200k+ LOC)
- No developer knows all parts
- Original authors are not in the team
- In development for 2+ years
- In production for 3+ years


CONSTITUTION CLASS STARSHIP U.S.S. ENTERPRISE NCC-1701
PORT SIDE ELEVATION/PROFILE VIEW MID-LINE INTERNAL ARRANGEMENT

TRASYS
WE GET IT DONE

Think the future today



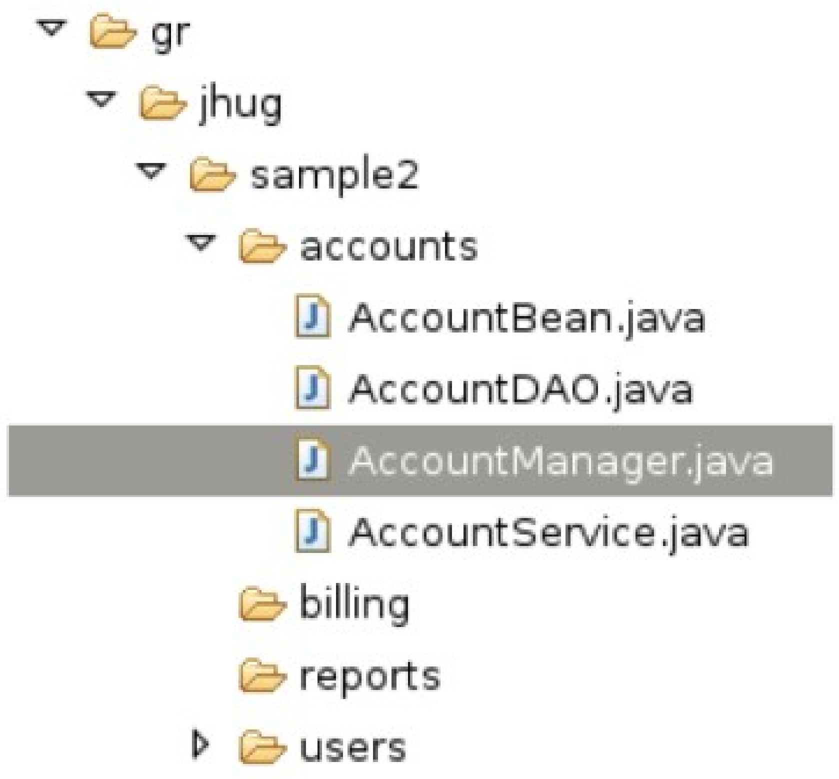
Numbers vary from 60% to 80%

This talk is about

Organization of Java packages by feature.

…and not by layer

Package by layer



- Package according to business purpose
- All relevant classes inside.

TRASYS WE GET IT DONE

Agenda – (the maintenance nightmare)

1. Clients always request features (not layers)
2. Encapsulation (follow the OOP paradigm)
3. Enforcing a sound software architecture
4. Plugin system (lego development)
5. Project code should grow horizontally (feature scope)

TRASYS WE GET IT DONE

Part I - Clients think in features

“There is a bug in the billing application”

Clients think in features

“This report has errors”

TRASYS WE GET IT DONE

Clients think in features

“The budget screen is missing a button”

Clients think in features

“When I save a document I get an error”

TRASYS
WE GET IT DONE

Clients think in features (never in layers)

~~“I think we should use AOP in our persistence layer”~~

Not!

A real story

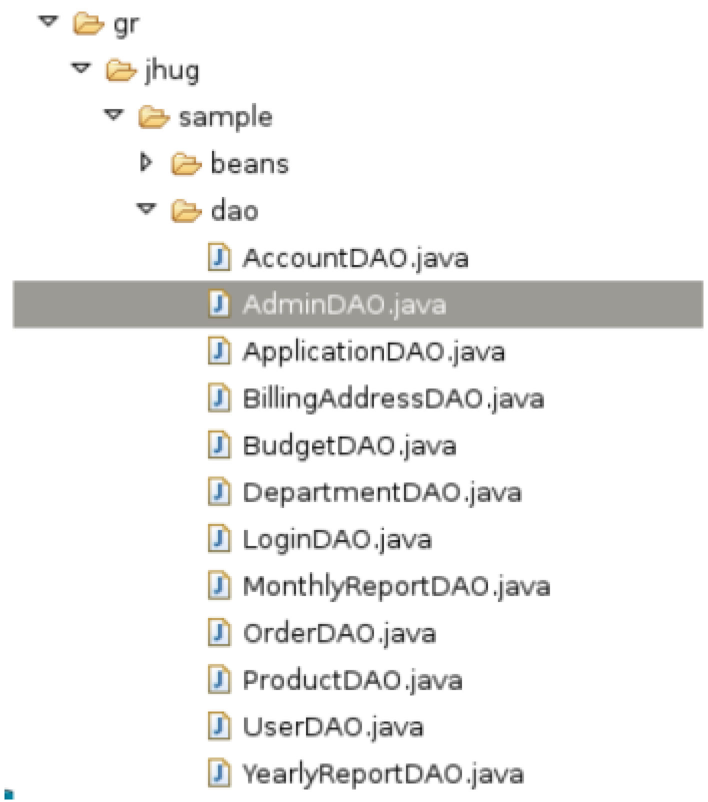
…with a software company this time

TRASYS
WE GET IT DONE

A real story

- Enthusiastic software developer
- Goes to a new company!
- Assigned to a project
- Gets his first issue to work on!
- “The profits report has a math error”

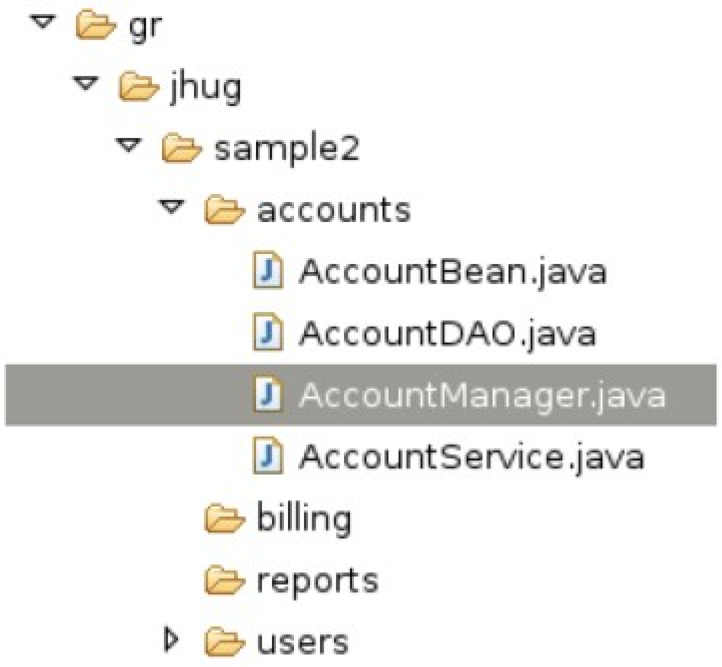
Is this easy?



```
gr
  jhug
    sample
      beans
      dao
        AccountDAO.java
        AdminDAO.java
        ApplicationDAO.java
        BillingAddressDAO.java
        BudgetDAO.java
        DepartmentDAO.java
        LoginDAO.java
        MonthlyReportDAO.java
        OrderDAO.java
        ProductDAO.java
        UserDAO.java
        YearlyReportDAO.java
```

- Nothing makes any sense
- Don’t know where to start looking
- Usually you need to ask around

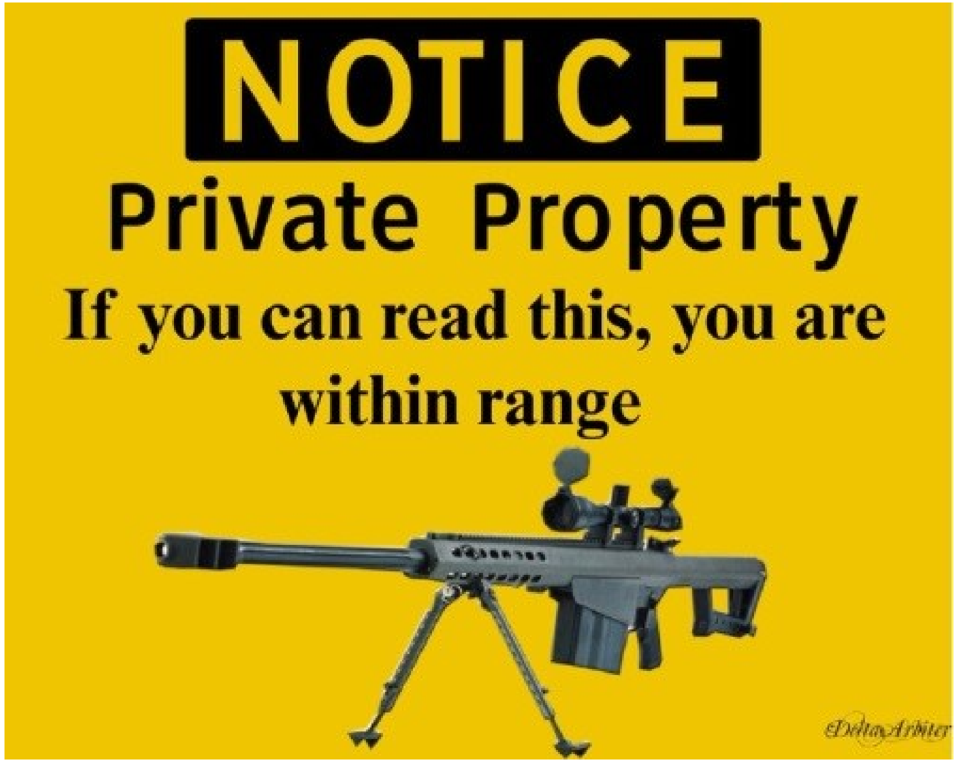
A better alternative



- Instant detection of affected code
- Changes contained in that package
- No need to look at the rest of the code
- Isolate junior developers

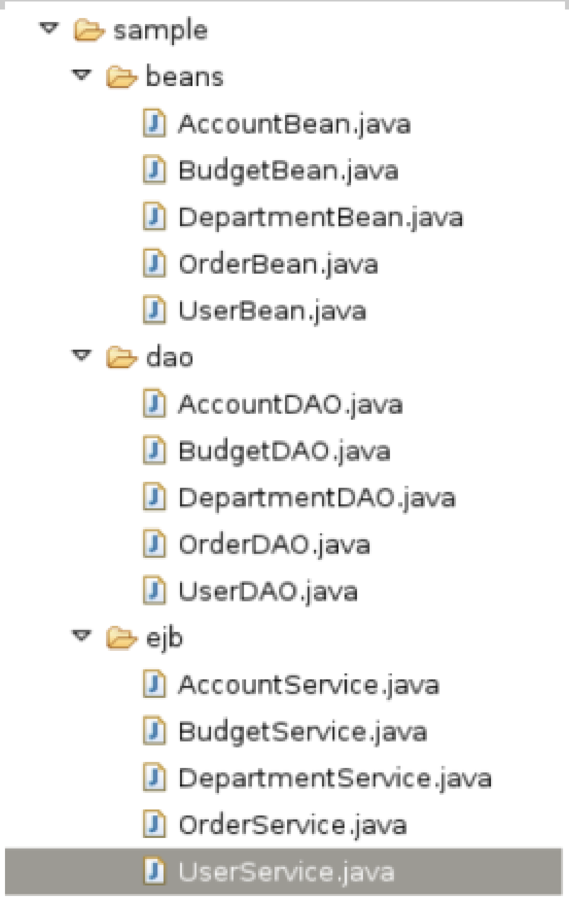
TRASYS WE GET IT DONE

Part II - Encapsulation



OOP in the package level

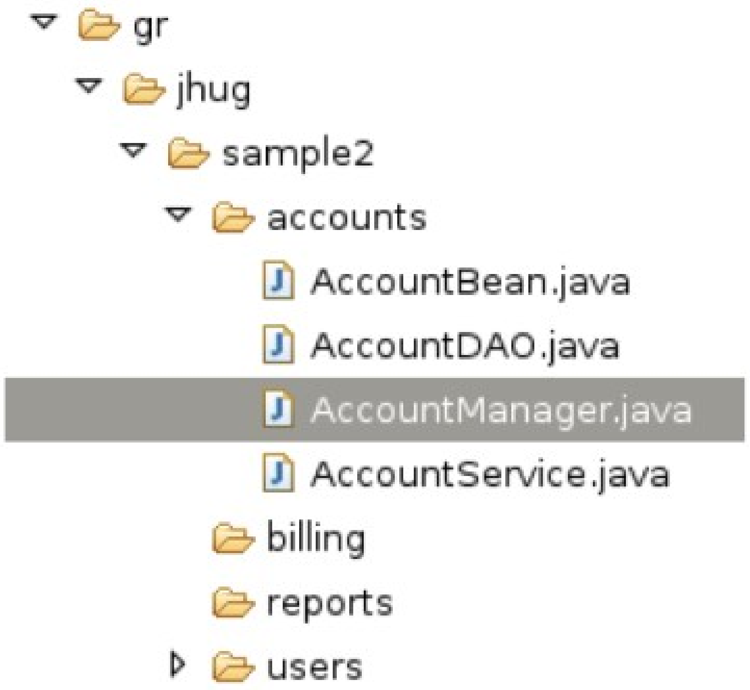
Part II - Encapsulation


```
sample
  beans
    AccountBean.java
    BudgetBean.java
    DepartmentBean.java
    OrderBean.java
    UserBean.java
  dao
    AccountDAO.java
    BudgetDAO.java
    DepartmentDAO.java
    OrderDAO.java
    UserDAO.java
  ejb
    AccountService.java
    BudgetService.java
    DepartmentService.java
    OrderService.java
    UserService.java
```

- All classes are public !
- Everything can be accessed by everything else

TRASYS
WE GET IT DONE

Part II - Encapsulation



- gr
  - jhug
    - sample2
      - accounts
        - AccountBean.java
        - AccountDAO.java
        - AccountManager.java
        - AccountService.java
      - billing
      - reports
      - users

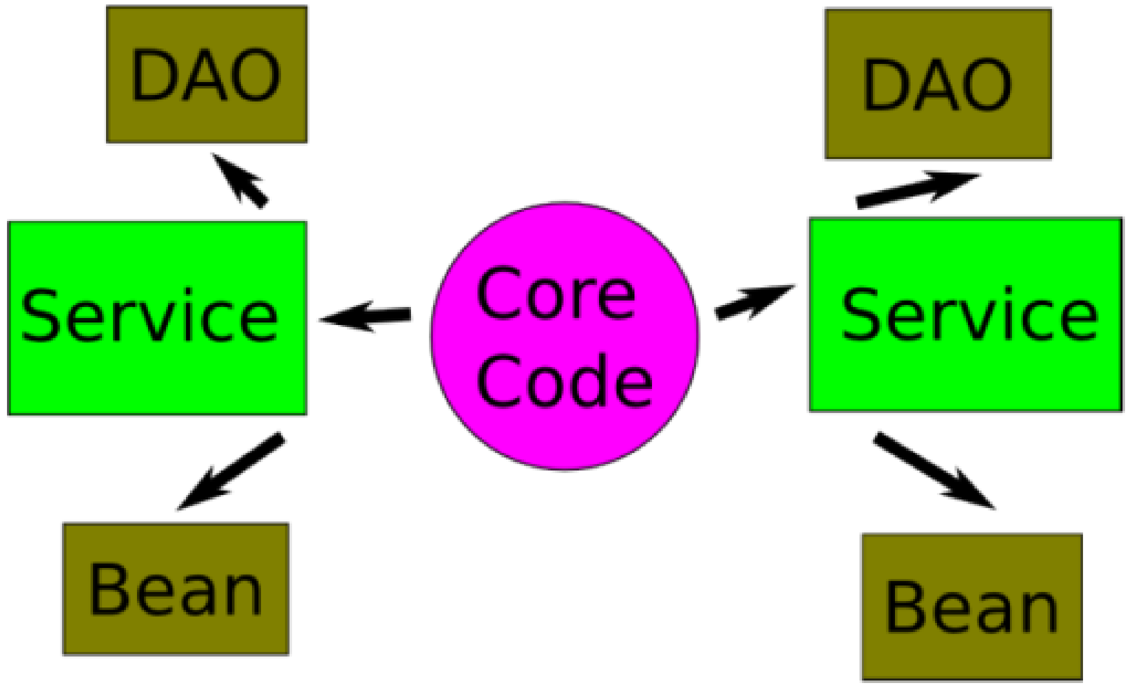
- DAO is package private
- Bean *could* be package private as well
- Only Service is public

Part III – Good architecture

Enforce a valid system design

TRASYS
WE GET IT DONE

Part III – Good architecture

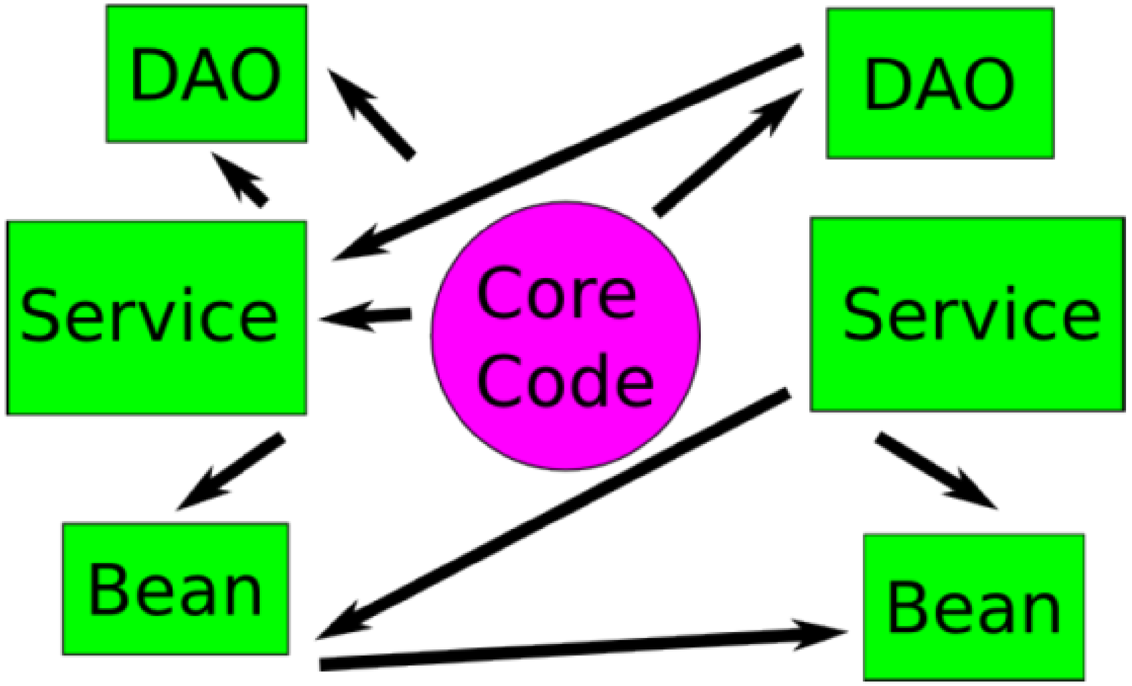


A good design

Part III – Good architecture

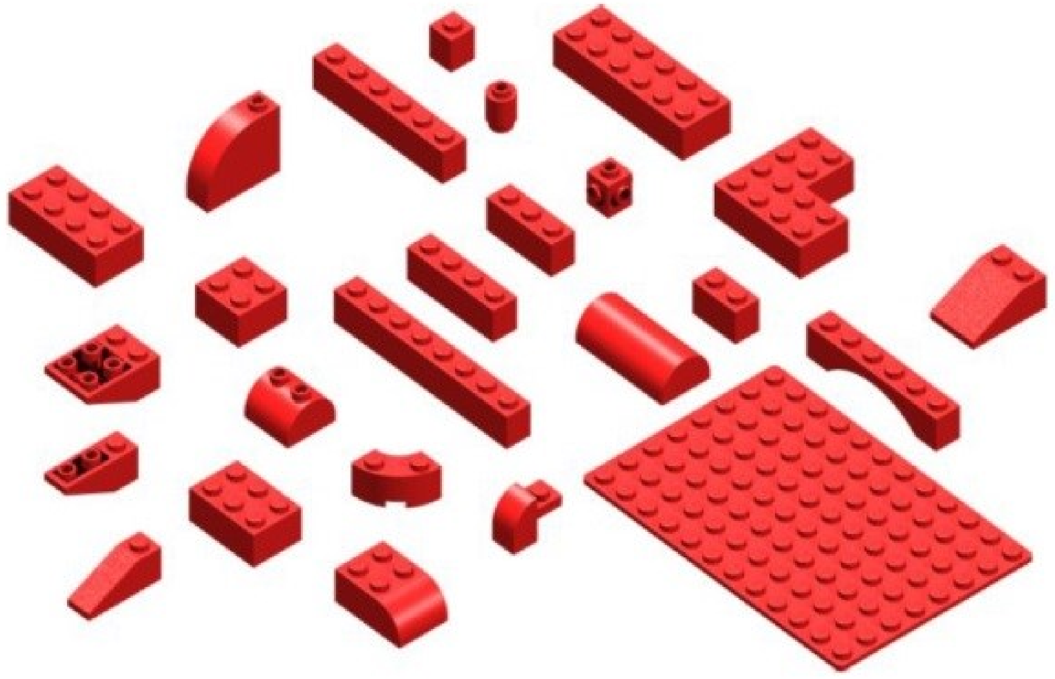
Package by layer can easily lead to …

Part III – Good architecture



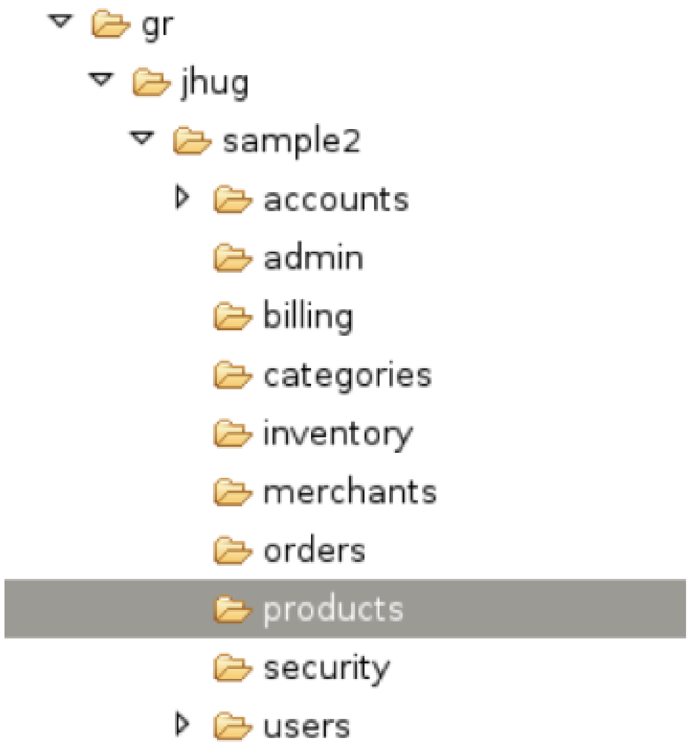
A bad design!

Part IV- Plugin structure



The Holy Grail of Enterprise applications

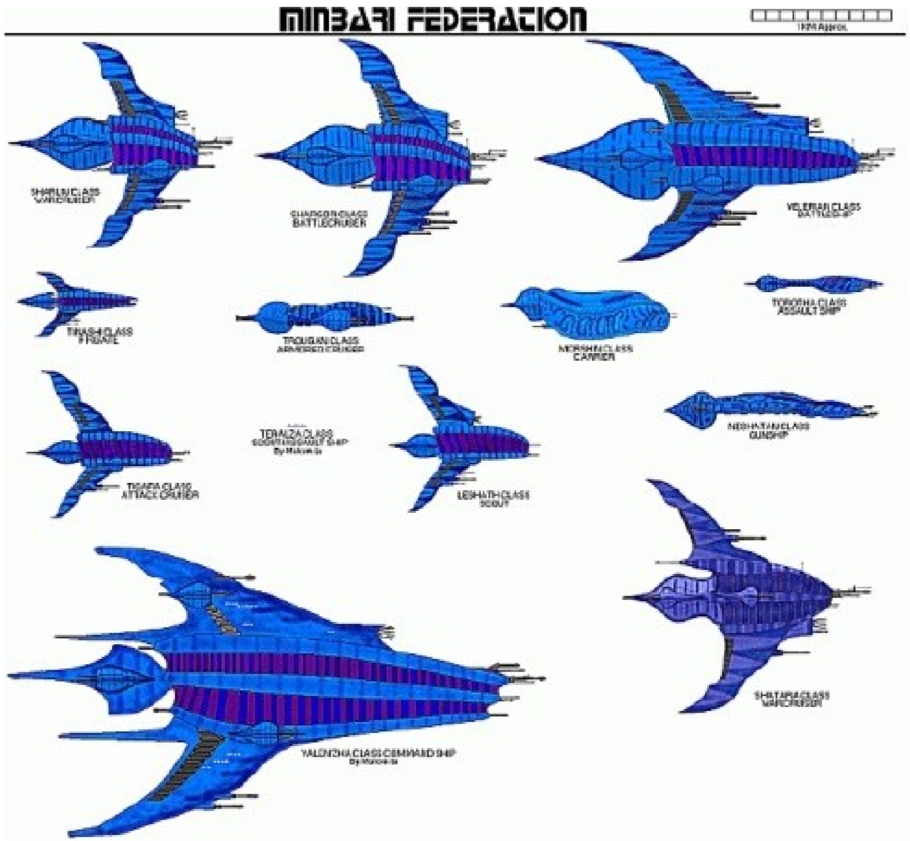
Part IV- Plugin structure



```
gr
  jhug
    sample2
      accounts
      admin
      billing
      categories
      inventory
      merchants
      orders
      products
      security
      users
```

- Packages are self-contained!
- They can be added in other projects
- They can be removed
- They can be converted to jars/wars/ears/OSGI etc.

Part V- Project size



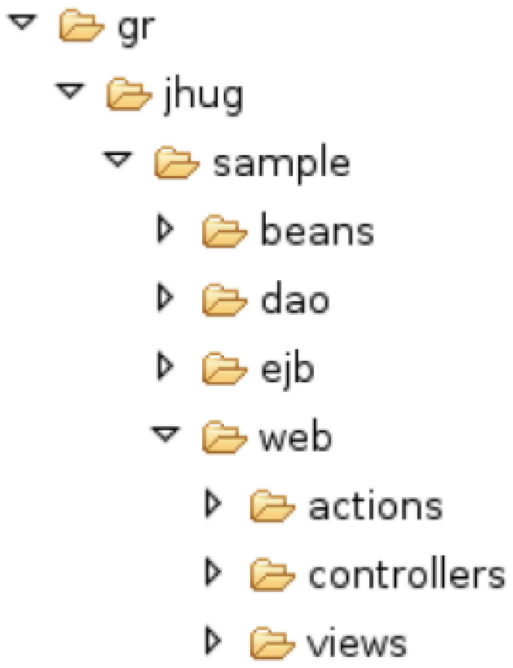
TRASYS
WE GET IT DONE

Part V- Project size

- Assume you have two enterprise projects
- The second could be just a newer version
- First project is 100.000 lines of code
- Second project is 1.000.000 lines of code
- How do they look in Eclipse?

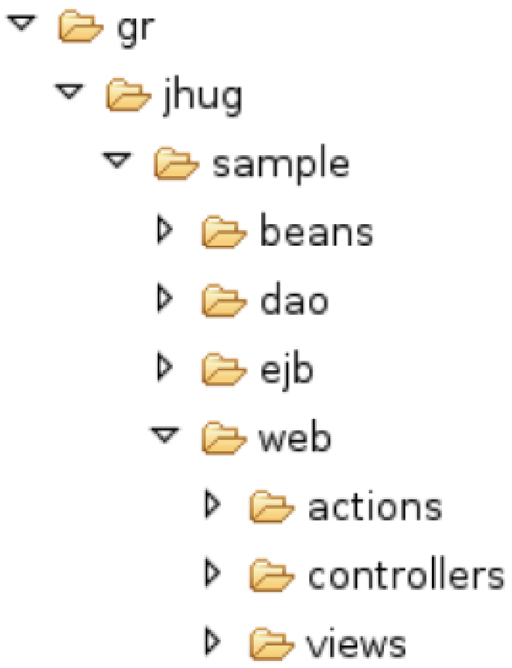
Package by Layer

Project 1



```
gr
  jhug
    sample
      beans
      dao
      ejb
      web
        actions
        controllers
        views
```

Project 2

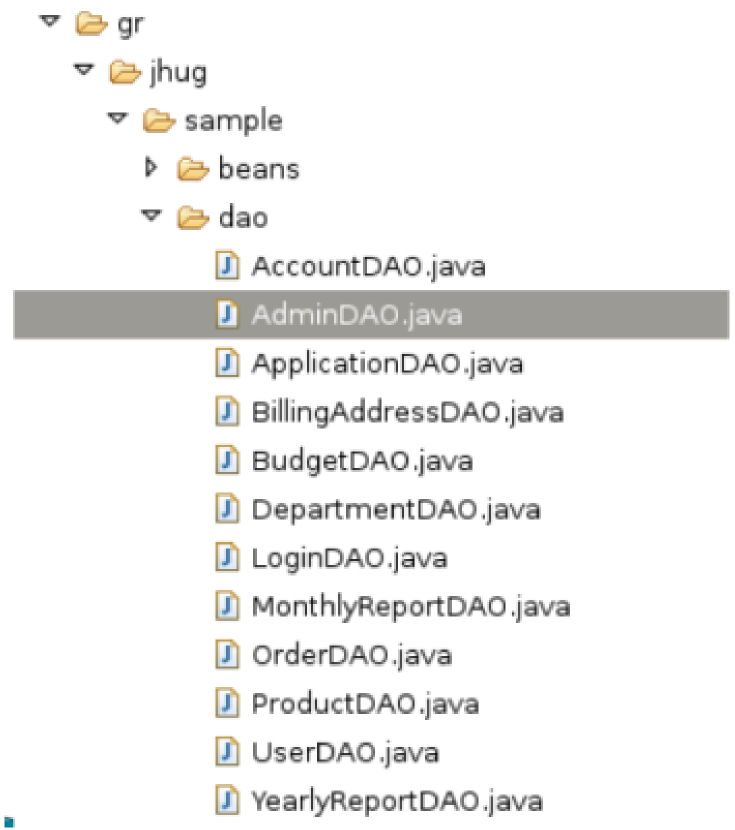


```
gr
  jhug
    sample
      beans
      dao
      ejb
      web
        actions
        controllers
        views
```

REJECTED

TRASYS
WE GET IT DONE

Package by Layer – A big project



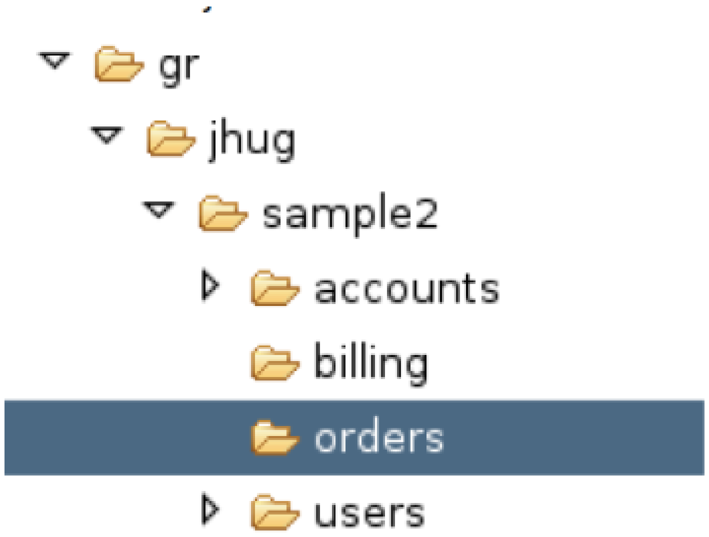
```
gr
  jhug
    sample
      beans
      dao
        AccountDAO.java
        AdminDAO.java
        ApplicationDAO.java
        BillingAddressDAO.java
        BudgetDAO.java
        DepartmentDAO.java
        LoginDAO.java
        MonthlyReportDAO.java
        OrderDAO.java
        ProductDAO.java
        UserDAO.java
        YearlyReportDAO.java
```

- Thousands of “actions”, DAOs
- Usually alphabetically sorted
- Very hard to work with
- Cause for code duplication

TRASYS
WE GET IT DONE

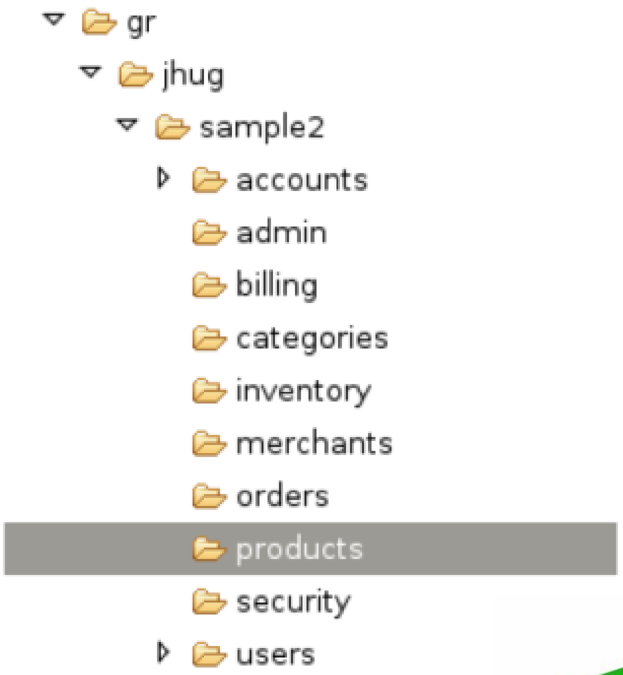
Package by feature

Project 1



- gr
  - jhug
    - sample2
      - accounts
      - billing
      - orders
      - users

Project 2



- gr
  - jhug
    - sample2
      - accounts
      - admin
      - billing
      - categories
      - inventory
      - merchants
      - orders
      - products
      - security
      - users

TRASYS
WE GET IT DONE

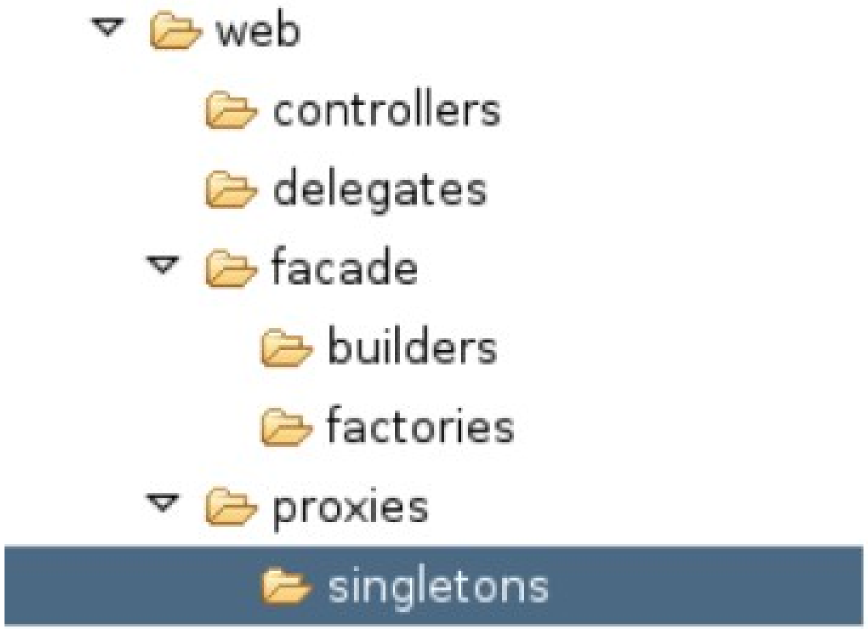
Closing remarks

Next time you add a new class to a package named:

- controllers
- dialogs
- actions
- DAO

Think Again!

Closing remarks



```
▽ web
    controllers
    delegates
  ▽ facade
      builders
      factories
  ▽ proxies
      singletons
```

- There is also a hybrid approach. First level is by feature and second layer is by layer
- Also avoid package by pattern (shown on picture)

Discussion

TRASYS WE GET IT DONE