OOA/OOD/OOP Example

example

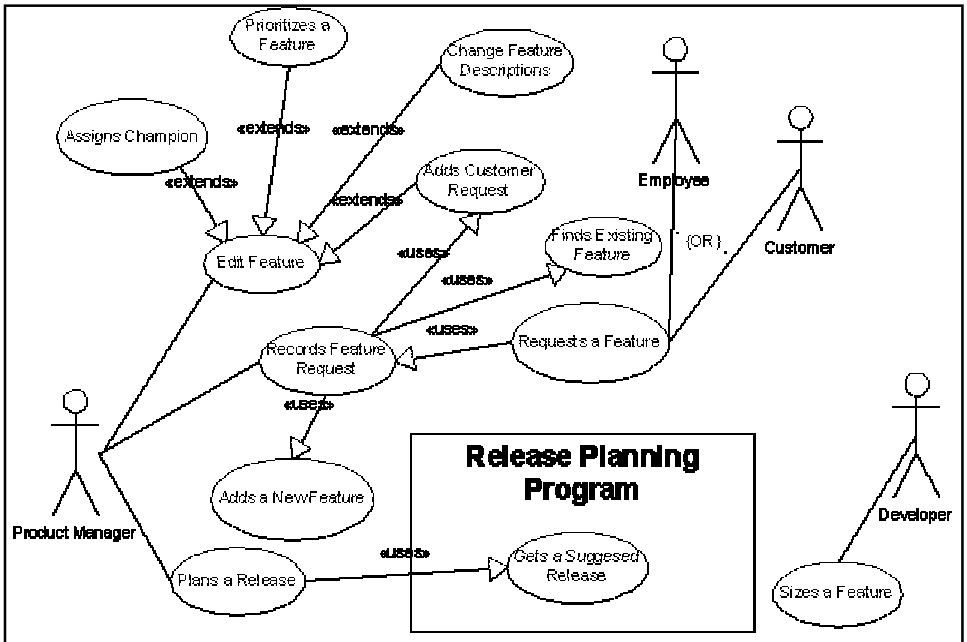
Requirements

- See eg/req.html
- Want a program to help a software company plan new releases of their software

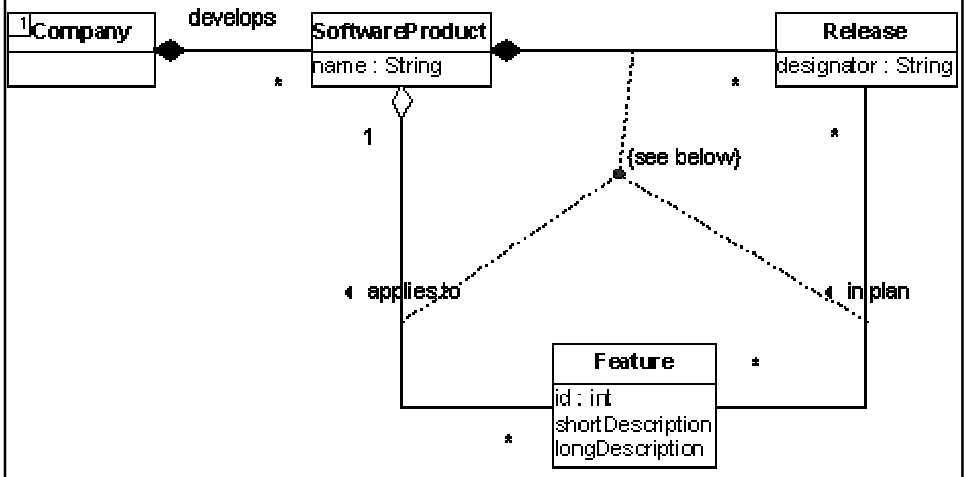
  java Plan features.xml Planetaria 340
- xml file contains sized (in coder days), prioritized (hi,med,low), feature requests for various products
  - includes list of requesting customers with how much they want it (1-10).
- Suggest an "optimal" release plan given the available capacity (in coder days).
- Sample output

OOA

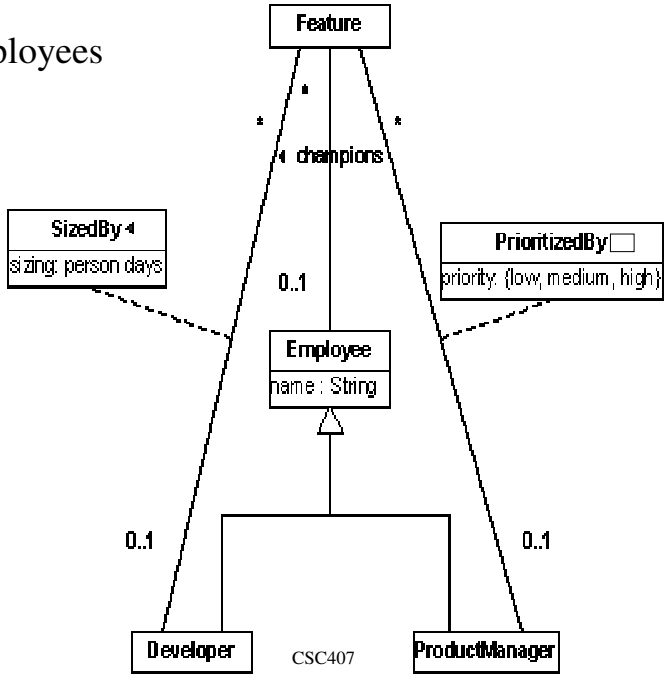
- See ooa document
- Introduction
  - why are we doing this
  - what is the current document for
  - where did the information come from
  - general points (change & XML file in this case)
- Use Cases
  - what is the bigger problem
  - how does this particular program fit into it
- Class Diagrams
  - restate information from the requirements statement in UML
  - (mostly you have no "requirements statement")



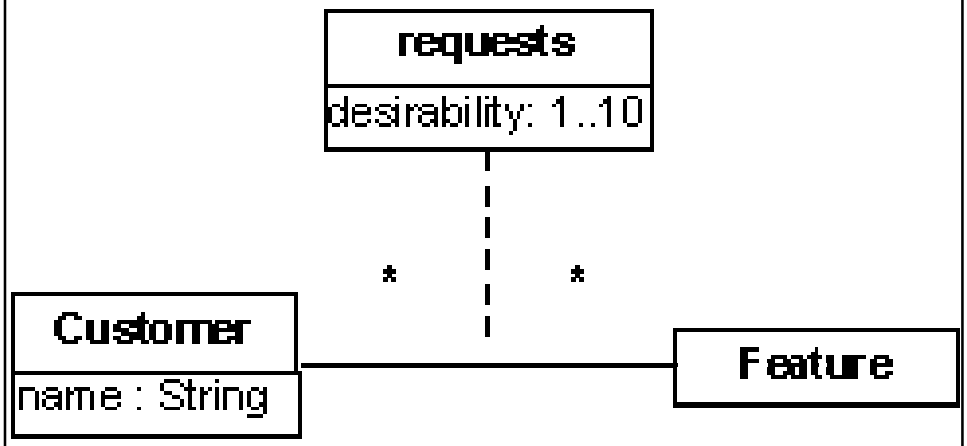
Features



Employees



Customers



OOD

- See ood document
- Package design
  - what rationale for the package breakdown
- Main driver
  - sequence diagram explaining how (one) use case is executed
- For each package
  - a collection of class diagrams
    - shows important methods
    - shows important attributes
    - shows association navigability
    - indicates how associations are implemented
    - indicates inheritance and interface implementation

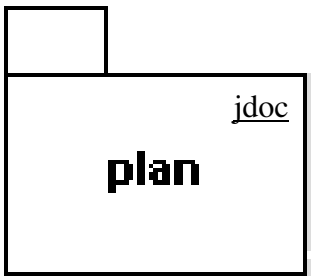
important = helps in understanding the design

About Source and Javadoc

- Javadoc is a tool that extracts comments formatted in a certain manner and produces Web pages documenting the details of a class design.
  - See example
- To display source code, I used a tool called java2html for pretty-printing Java source to HTML.
  - See example

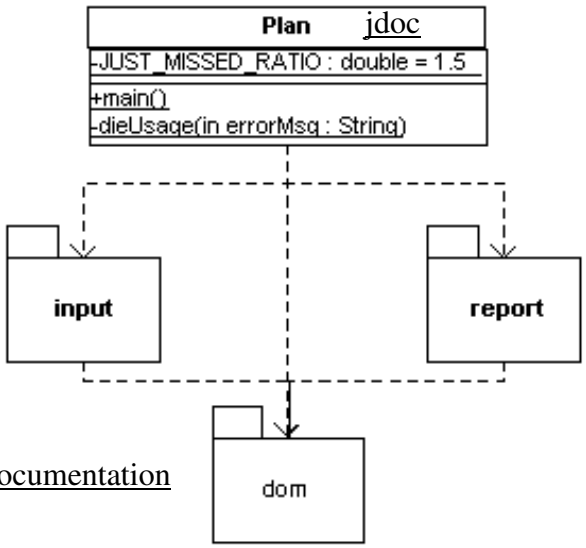
ood

Top Package

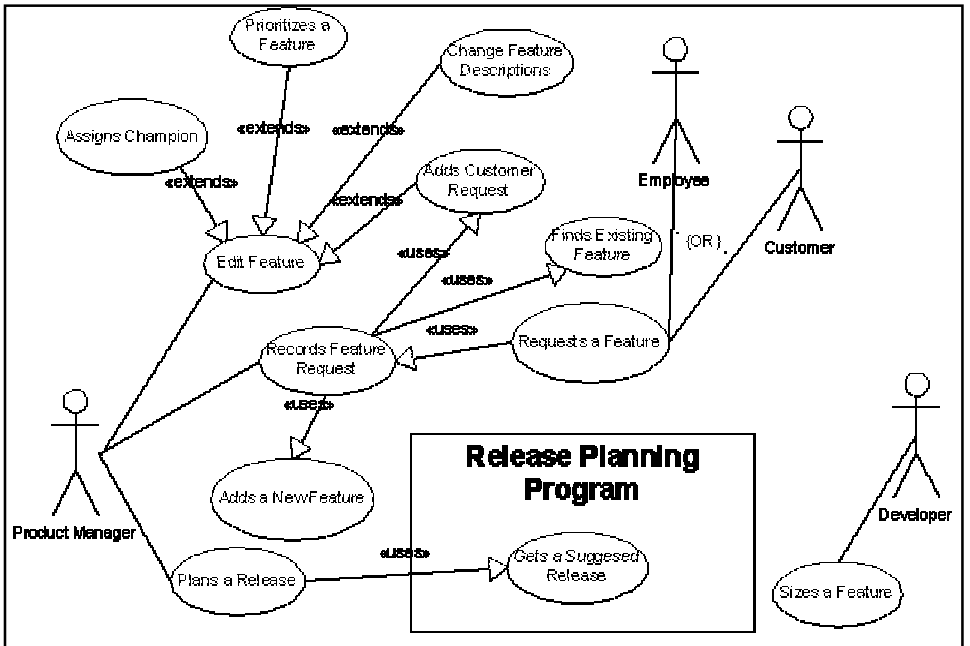


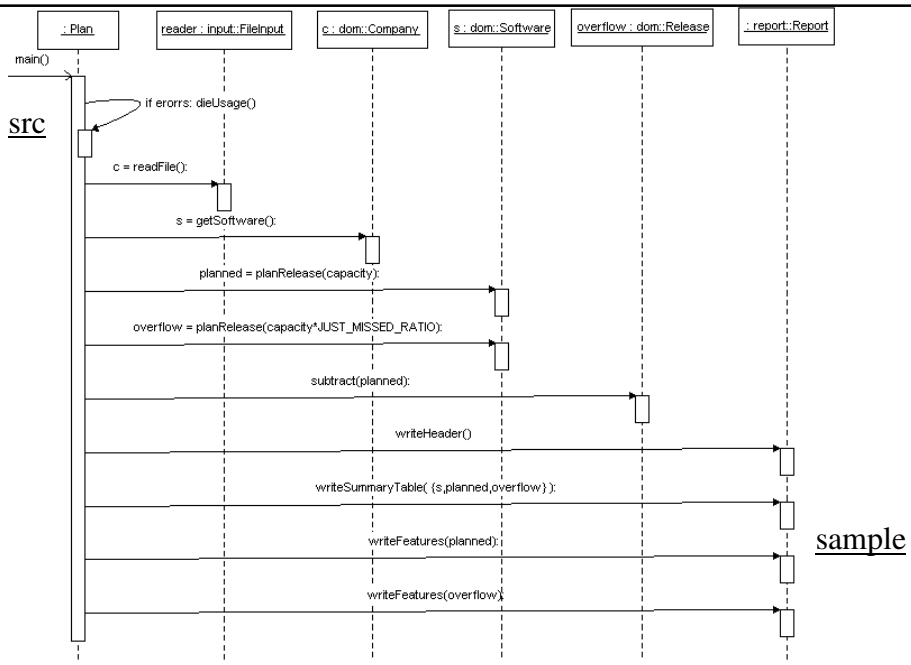
ood

Package Plan



package documentation



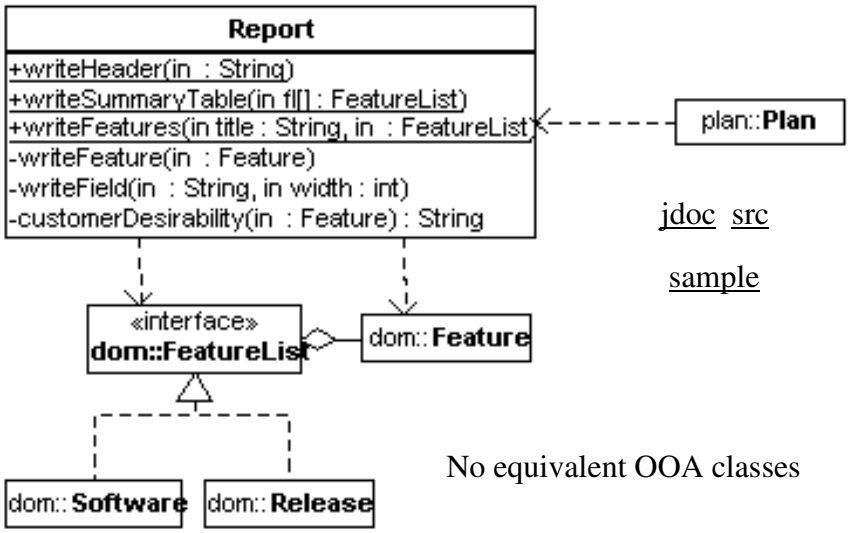


src

sample

ood

Package Report



jdoc src

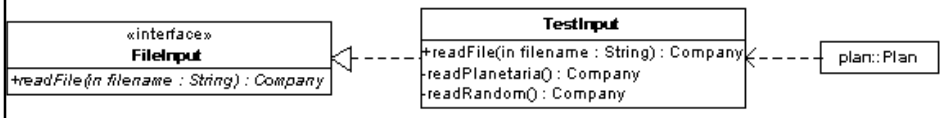
sample

No equivalent OOA classes

ood

Package Input

jdoc src



- No equivalent OOA classes
- Sequence diagram for readFile is fairly clear just from the class description (see also Report class)

ood

Package dom

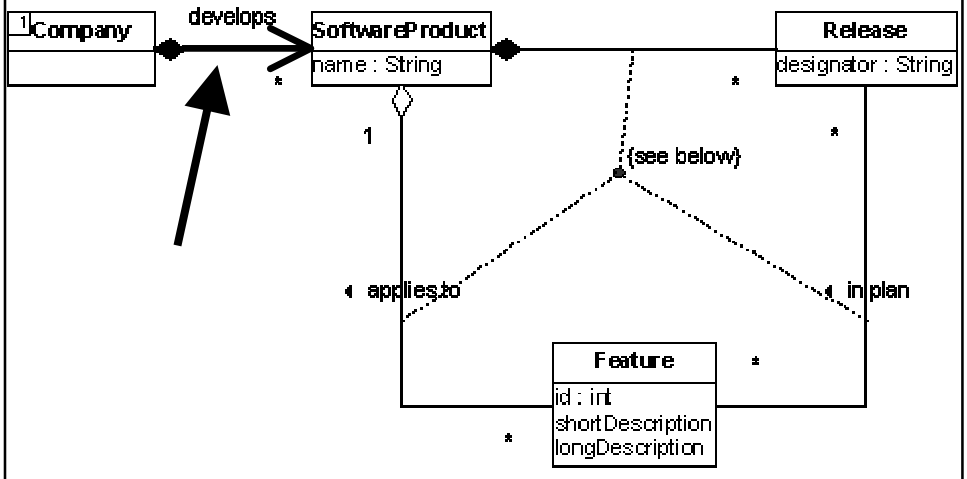
- For "Domain Object Model"
- Coad's "Problem Domain Component"
- Implements an in-memory, object-oriented data model reflecting the OOA
- Must be modified/extended to work in a program

jdoc

Implementing Associations

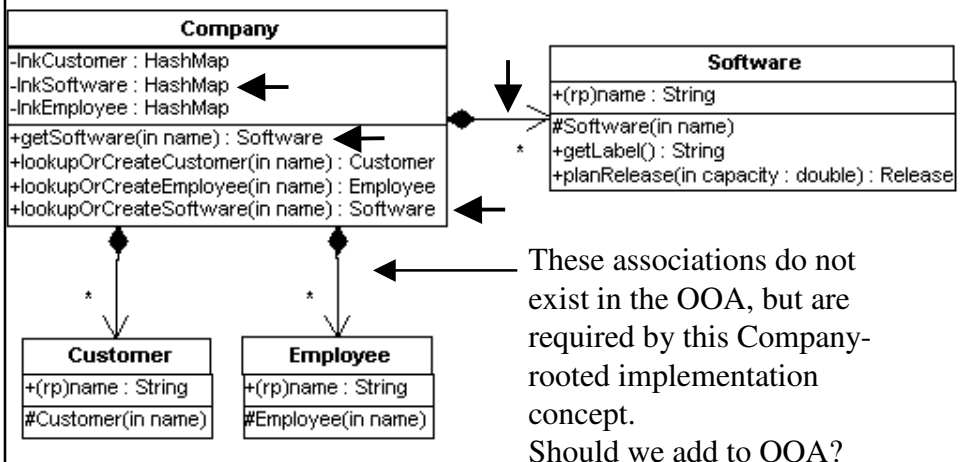
- Decide on navigability
  - The direction in which the association can be efficiently navigated
    - If you have one object of the Left class, can you in O(n) time access all objects of the Right class linked to that Left object.
- Decide on interface for
  - Navigating the links
    - usually get method for 1 side, iterator for * side.
  - Adding new links
  - Deleting links (if necessary)
- Decide on implementation
  - Simple pointer to implement the [0..1] side
    - (if required by navigatability)
  - Array, Vector, Map, Linked List to do the [*] side
    - (if required by navigatability)

Features (from OOA)



ood jdoc src

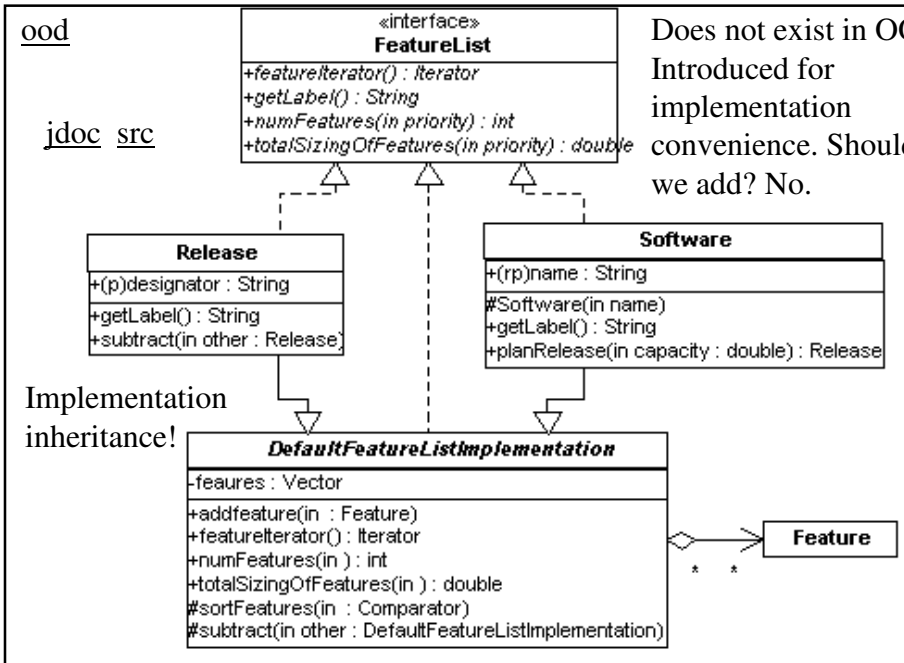
DOM Company



These associations do not exist in the OOA, but are required by this Company-rooted implementation concept.
Should we add to OOA? Maybe.

ood

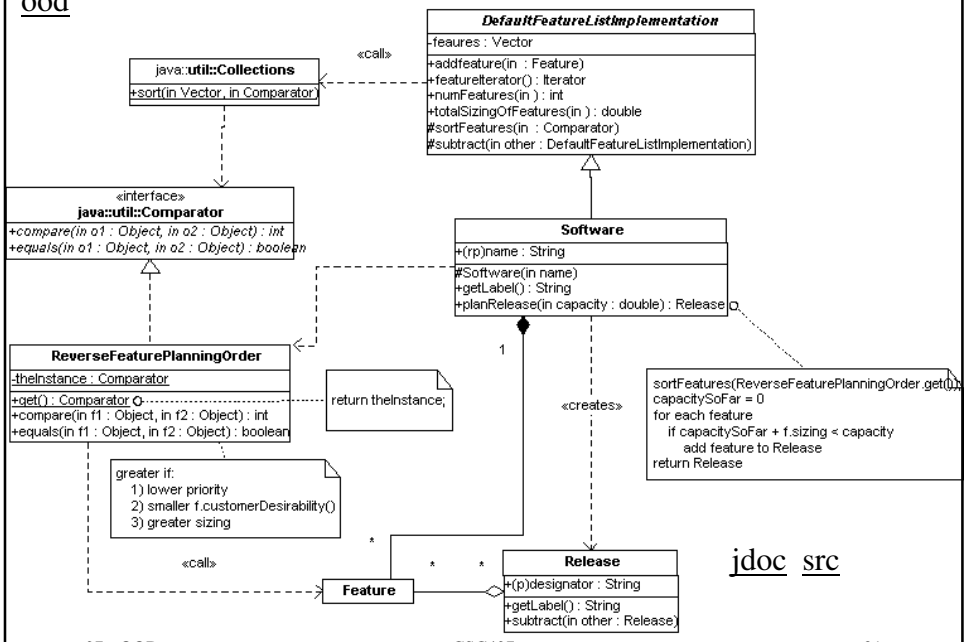
jdoc src



Does not exist in OOA. Introduced for implementation convenience. Should we add? No.

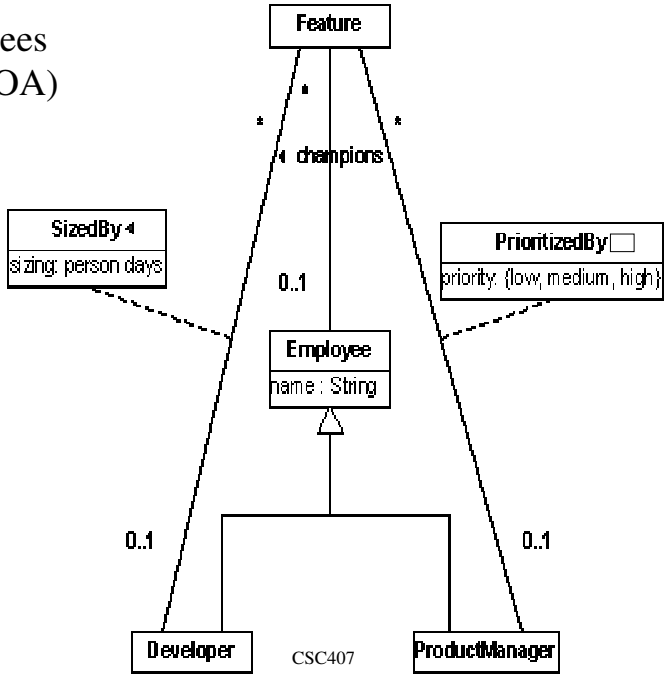
Implementation inheritance!

ood

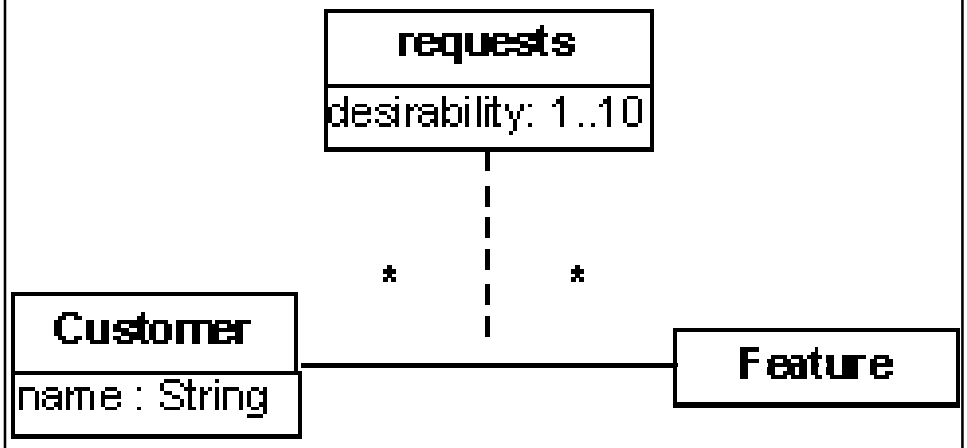


jdoc src

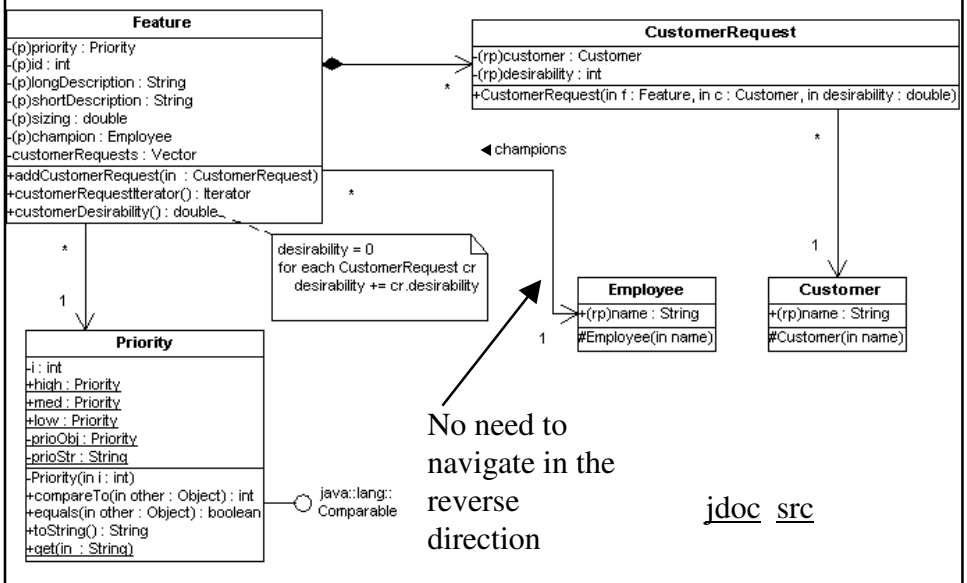
Employees (from OOA)



Customers (from OOA)



ood



No need to navigate in the reverse direction

jdoc src