

CS 535 Object-Oriented Programming & Design Spring Semester, 2003 Doc 19 OO Design - Analysis Phase Contents

Contents.....	1
References	1
Overview of Design Process.....	2
Analysis Phase.....	3
Hierarchies.....	4
Building Good Hierarchies.....	5
Identifying Contracts	7
Subsystems.....	9
Collaboration Graphs.....	12
Identifying Subsystems	14
Protocols.....	17
Refining Responsibilities.....	18
Specifying the Design.....	21

References

Designing Object-Oriented Software, Wirfs-Brock, chapters 6 - 8

Copyright ©, All rights reserved. 2003 SDSU & Roger Whitney, 5500 Campanile Drive, S. 92182-7700 USA. OpenContent (<http://www.opencontent.org/opl.shtml>) license defines the this document.

Overview of Design Process

Exploratory Phase

Finding the objects

Determining responsibilities

Finding collaborations

Analysis Phase

Finding hierarchies

Finding subsystems

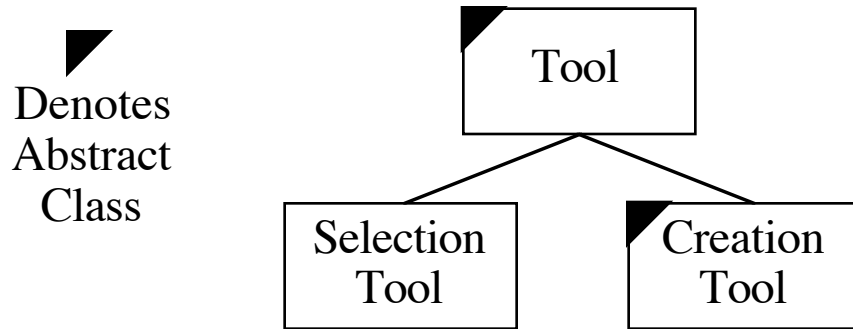
Refining the design

Analysis Phase

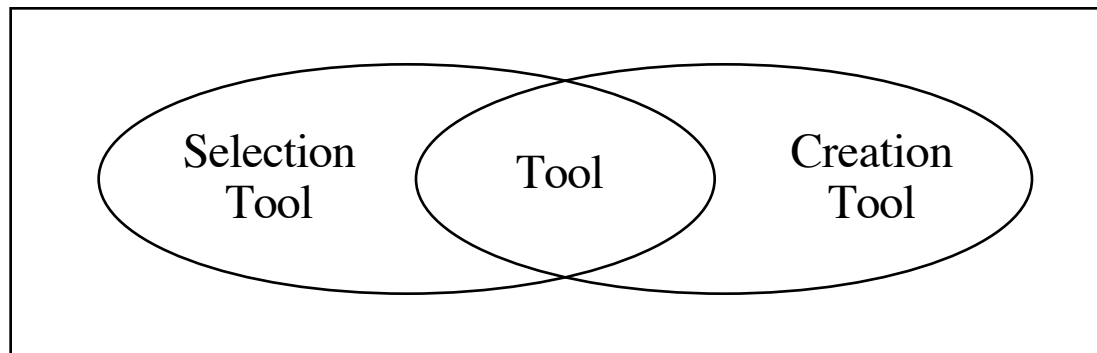
- Finding Inheritance
 - Determine which classes are related via inheritance
 - Finding abstract classes
 - Determine class contracts
- Finding Object Interaction
 - Divide responsibilities into subsystems
 - Designing interfaces of subsystems and classes
- Refining the Design
 - Construct protocols for each class
 - Produce a design specification for each class and subsystem
 - Write a design specification for each contract

Hierarchies

Hierarchy Graphs

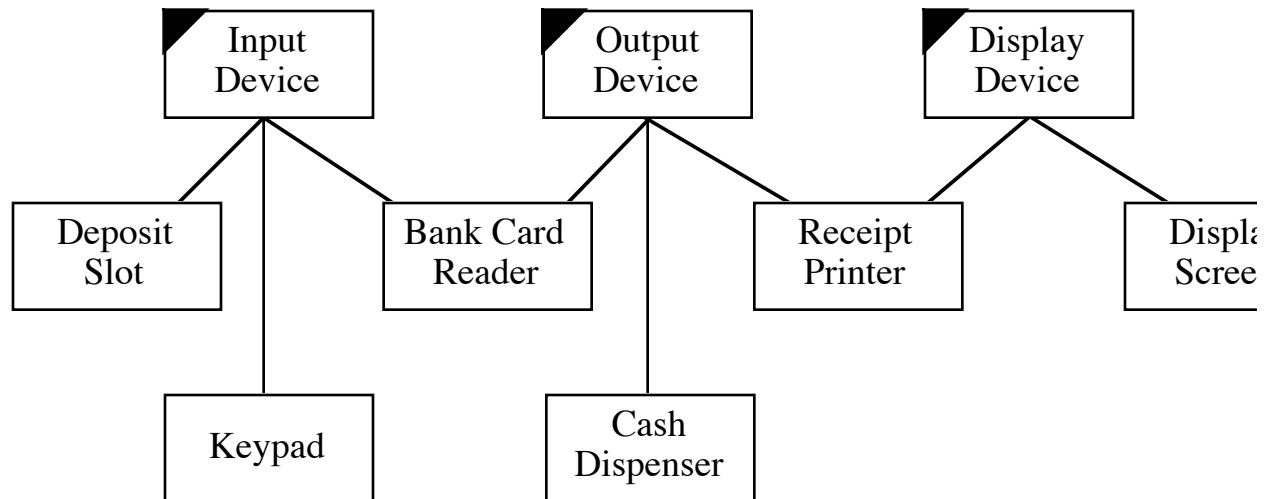


Venn Diagrams



Building Good Hierarchies

Model a "kind-of" hierarchy

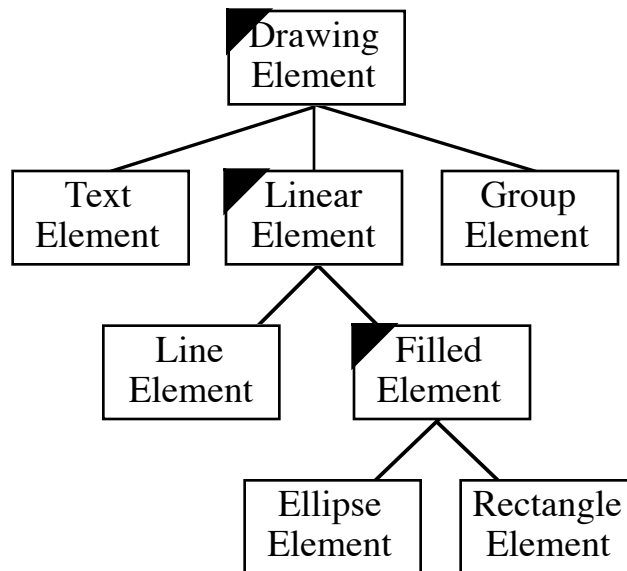
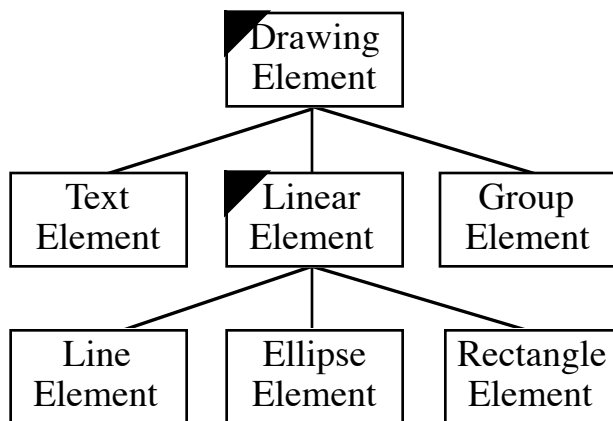
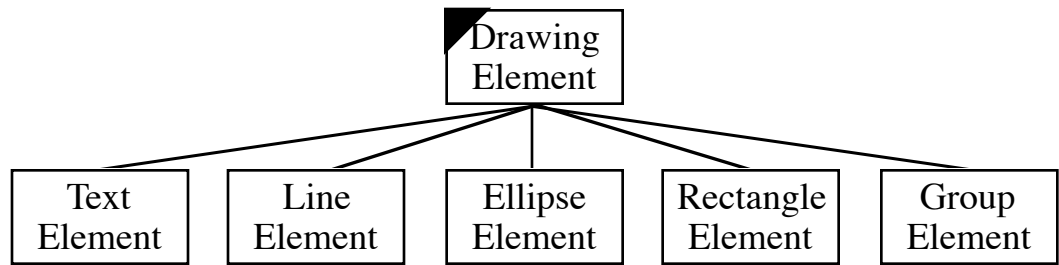


Multiple inheritance can be used in the design even if you use an implementation language inheritance.

Make sure that abstract classes do not inherit from concrete class

Eliminate classes that do not add functionality

Factor common responsibilities as high as possible



Identifying Contracts

Contract

Set of requests that a client can make of a server

Cohesive set of responsibilities that a client can depend on

Abstraction of a set of responsibilities of a class

Example: Account Class

Contract: Access and modify the account balance

Responsibilities:

- Know the account balance

- Accept deposits

- Accept withdrawals

Identifying Contracts

Group responsibilities used by the same clients

Maximize the cohesiveness of classes

Contract of a class should make sense together

Minimize the number of contracts

Use inheritance

Set of classes all supporting a common contract should inherit the contract from a common superclass

Applying the Guidelines

Start defining contract at the top of the hierarchies

Name and number each contract

For each collaboration, determine which contract represents that collaboration

Subsystems

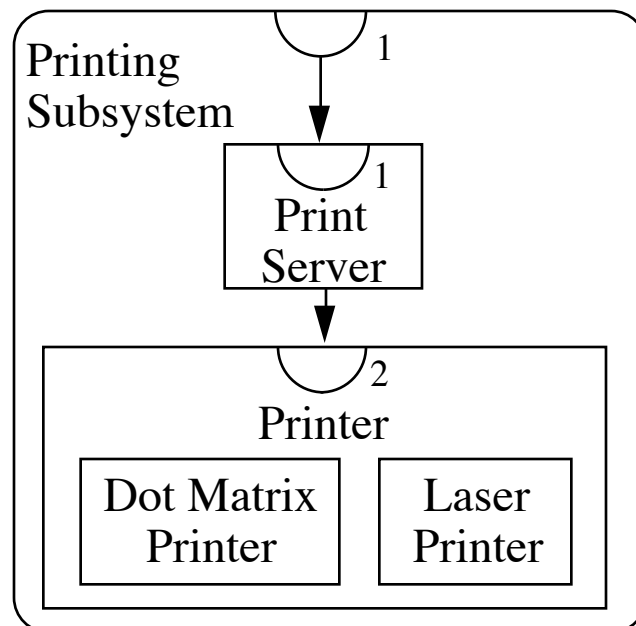
Subsystems are groups of classes, or groups of classes and other subsystems, that collaborate among themselves to support a set of contracts

There is no conceptual difference between the responsibilities of a class and a subsystem of classes

The difference between a class and subsystem of classes is a matter of scale

A subsystem should be a good abstraction

There should be as little communication between different subsystems as possible



Top-Down, Bottom-Up Large Systems

Most texts illustrate OO design "bottom-up"

- Find objects

- Determining responsibilities

- Determine object collaboration

- Find hierarchies

- Determine subsystems

Large systems are designed "top-down"

- Find top level subsystems

- Determine subsystem responsibilities

- Determine subsystem collaboration

- Find hierarchies

- Iterate above steps on each subsystem

Each level is built "bottom-up"

Levels are done "top-down"

Top-Down, Bottom-Up Large Systems

Jacobson, 1991

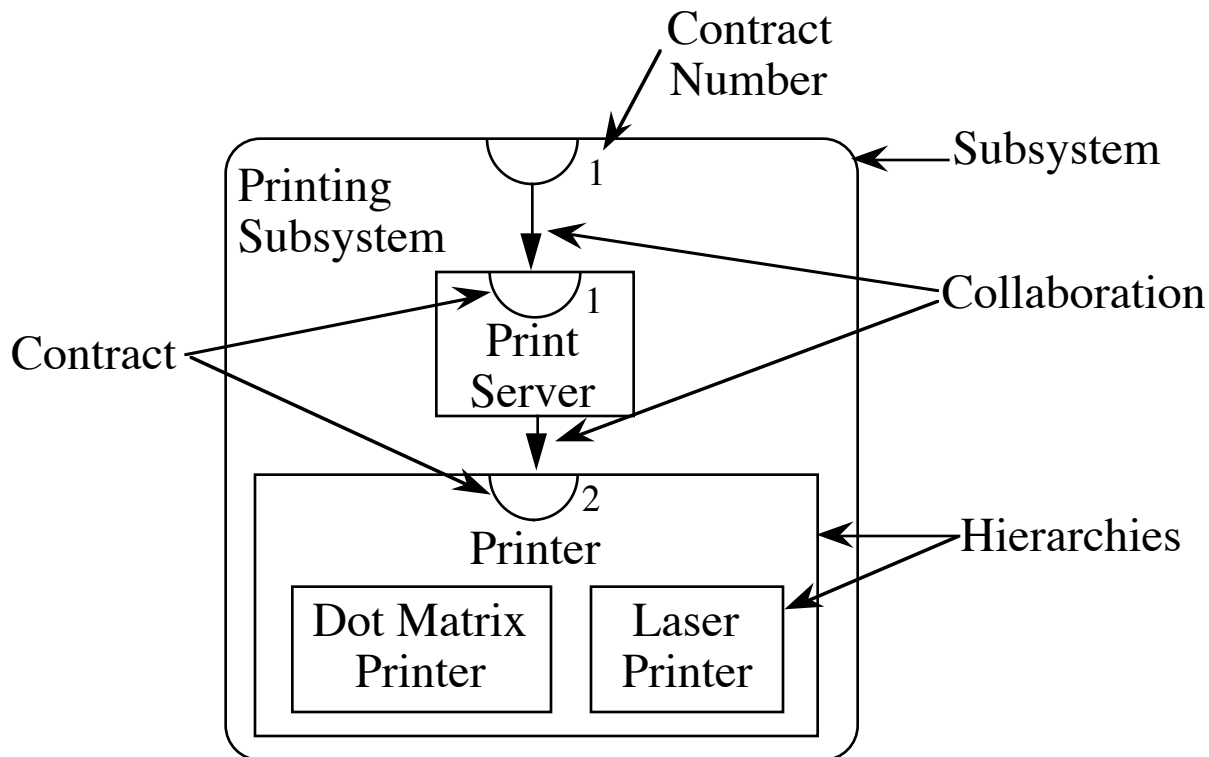
"The subsystem division in small projects is normally made at the end of the analysis, when the architecture is clear. In larger projects, it often must be done earlier, in many cases even before the analysis has been developed."

"In large systems it is often essential to develop the system in parallel."

"For large projects there may be other criteria for subsystem division, for example:

- Different specialties in different development groups
- If an existing product is to be used in the system, it may be regarded as a subsystem
- In a distributed environment, a subsystem may be wanted at each node"

Collaboration Graphs

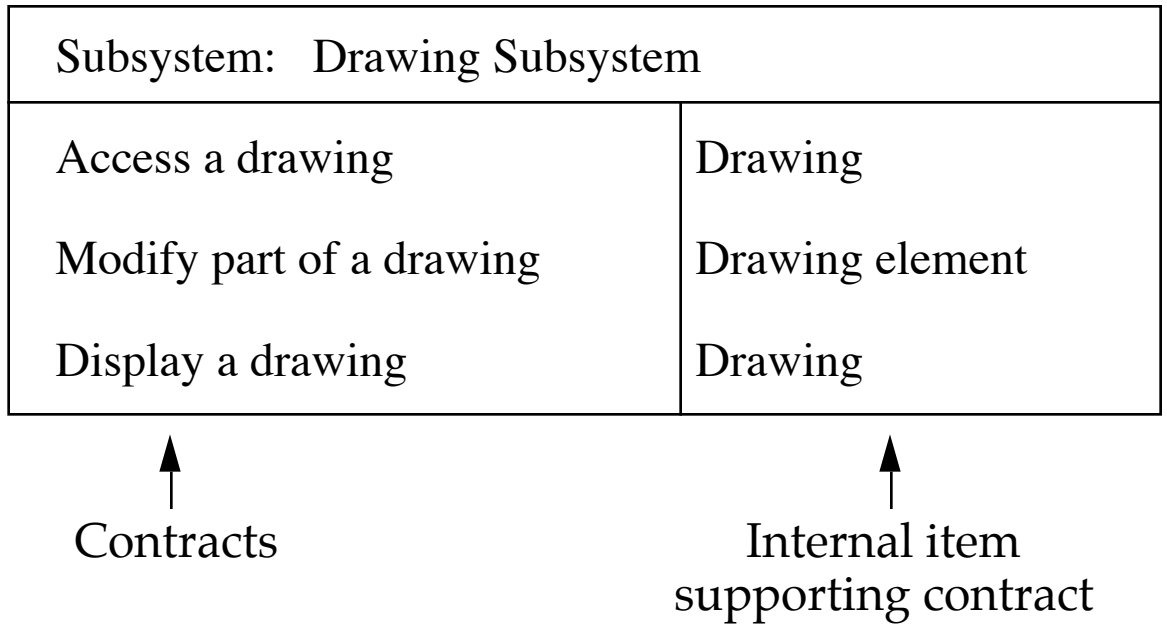


Subsystem Contracts

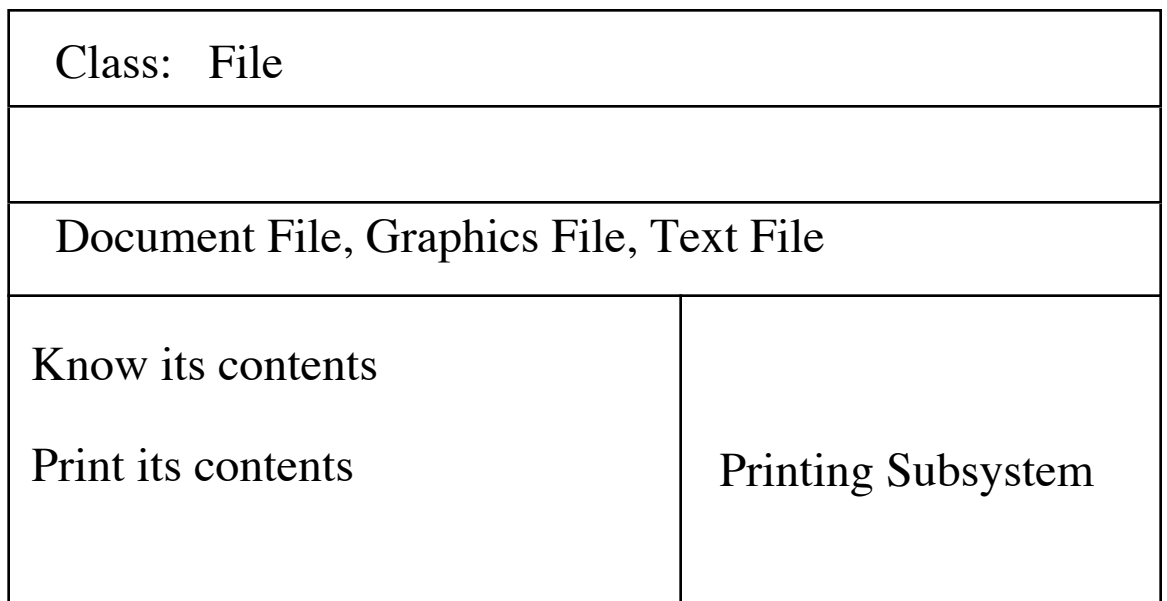
A subsystem contract consists of all class contracts that provide services to clients outside the system

Subsystem contracts can be extended

Subsystem Cards



Class Cards



Identifying Subsystems

All objects which have strong coupling should be placed in the same subsystem

There should be as little communication between different subsystem possible

Does a set of classes make sense as an abstraction?

Can you name a group of classes?

Does a group of classes interact frequently?

Simplifying Interactions

Subsystems

Reduce complexity of a design

Provide coherent structure to the design

Minimize the number of collaborations a class has with other classes and subsystems

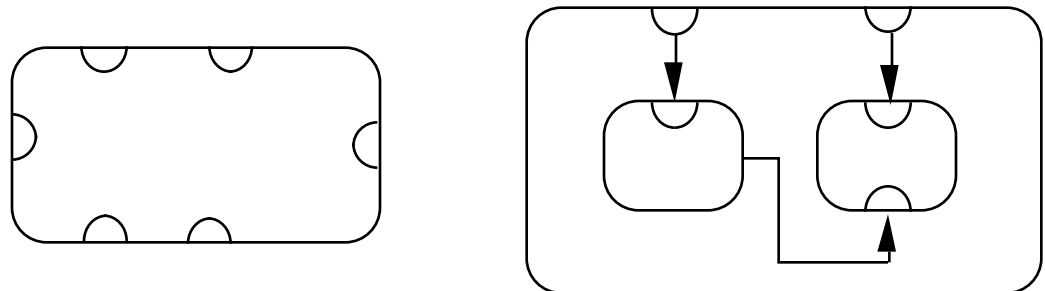
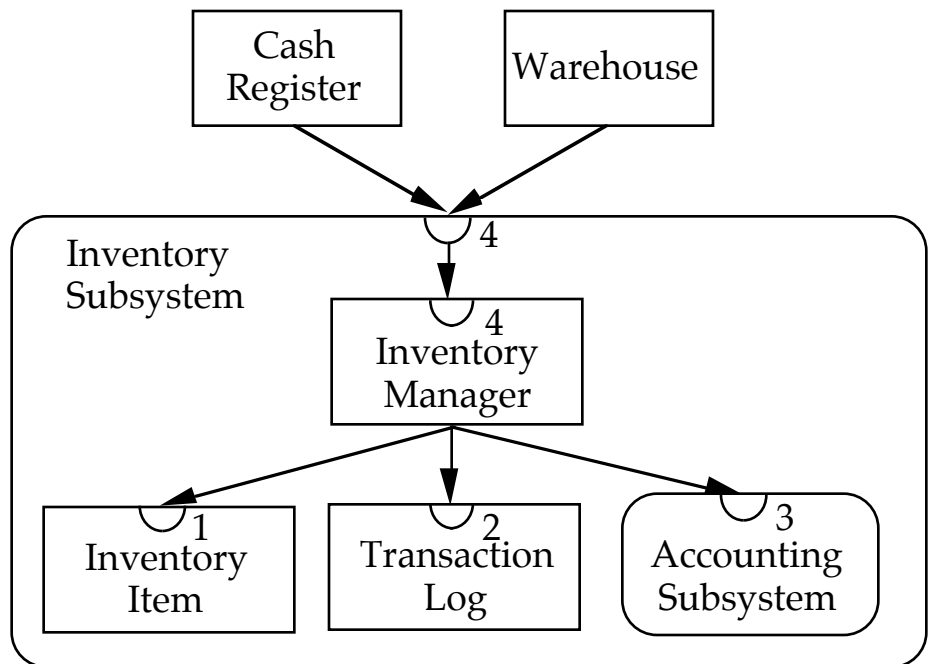
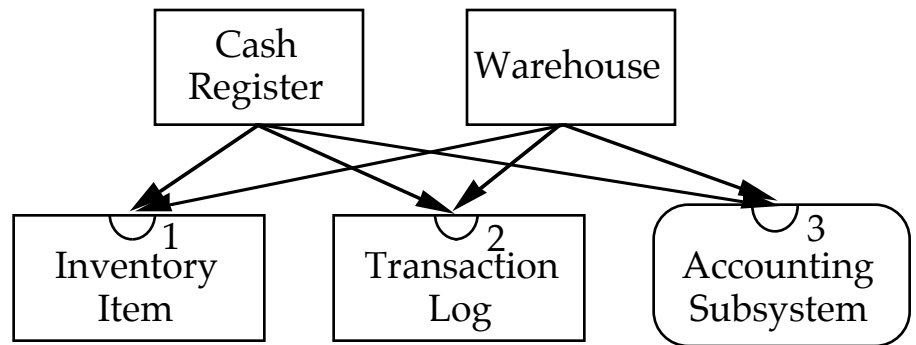
Reassign responsibilities or expand the knowledge of another class to create fewer collaborations

Create subsystem to reduce collaborations

Minimize the number of classes and subsystems to which a subsystem delegates

Minimize the number of different contracts supported by a class or subsystem

Too many contracts in one subsystem can be a sign that the subsystem has too much intelligence



Protocols

Construct protocols for each class

Specify the signatures for the methods that each class will im

Write a design specification for each class and subsystem

Write a design specification for each contract

Refining Responsibilities

Turn contracts into protocols

Account contract 1

Access and modify the account balance

Know the account balance

Accept deposits

Accept withdrawals

Protocols

balance() returns Fixed Point Number

deposit(Fixed Point Number)

withdraw(Fixed Point Number)

In general, private responsibilities represent designs notes to an implementers

Select operation names carefully

Don't use one name to mean two different things

Don't use two names for the same thing

Make protocols as generally useful as possible

Refining Responsibilities

Define reasonable defaults

First, define the most general message, one that allows client supply all possible required parameters

Next, provide default values for any parameter for which it is reasonable to do so

Finally, analyze how each client uses this general message. analysis, define a set of messages that allows clients to specify some of the parameters, while relying on defaults for the other

Refining Responsibilities

Define reasonable defaults

Example: Display of Drawing Elements

Parameters

Display device – printer or screen

Display region – clipping region

Drawing rule – how to combine new bits with old

Transformation – from element space to display space

Defaults

Display device – active window

Display region – entire medium

Drawing rule – over, completely replace old bits

Transformation – identity

Protocol

display()

display(Display Device)

display(Region)

display(Display Device, Region)

display(Display Device, Region, Drawing Rule)

display(Display Device, Region, Drawing Rule, Transformati

Specifying the Design Classes

Class: Drawing (Concrete)

Superclasses: Displayable Object

Subclasses: none

Hierarchy Graphs: page 5

Collaborations Graph: page 8

Description: This class represents the structure of ...

Contracts

1. Display itself

This contract is inherited from Displayable Object

2. Maintain the elements in a drawing

Know which elements are contained in the drawing

addElement (Drawing Element)

uses List

This method adds a drawing element ...

elementAt (Point) returns Drawing Element

uses List, Drawing Element (3)

This method returns the first drawing ...

Specifying the Design Classes

- Write the class name and state whether the class is abstract or concrete. List its immediate superclasses and subclasses
- Provide class's position in the hierarchy and collaboration graphs
- Describe the purpose of the class and its intended use
- List each contract for which the class is a server
- For each contract, list the responsibilities of the class that support it. Under each responsibility, write the signatures of the methods that implement the responsibility. Include a brief description and note the collaborations required. Don't neglect error conditions; specify the behavior of the method for all given inputs.
- List the private responsibilities that have been defined
- Include other relevant information:
 - behavioral constraints
 - implementation considerations

Specifying the Design Subsystems

Subsystem: Drawing Subsystem

Classes: Control Point, Drawing, Drawing Element, Ellipse Element, Group Element, Line Element, Linear Element, Rectangle Element, Text Element

Collaborations Graphs: pages 6 and 8

Description: The Drawing subsystem is responsible for displaying and maintaining the contents of a drawing. The Drawing Subsystem has three contracts. Two are supported by ...

Contracts

1. Display itself

This contract is defined by Displayable Object, and supported by Drawing

Server: Drawing

2. Access and modify the contents of a drawing

Server: Drawing

3. Modify the attributes of a Drawing Element

Server: Control Point

Specifying the Design Subsystems

- Write the subsystem name at the top of the page
- List all encapsulated classes and subsystems
- Provide subsystems position in the hierarchy and collaboration graph
- Describe the purpose of the subsystem
- List the contracts for which this subsystem is a server
- For each contract, identify the class or subsystem to which the contract is delegated

Specifying the Design Formalizing Contracts

Contract 3: Modify the attributes of a drawing element

Server: Control Point

Client: Selection Tool

Description: This contract allows modification of a drawing element through the manipulation of a control point associated with that element. The result of moving the control point is specified by the drawing element at the time the control point is created.

For each contract include:

Contract name and number

Server(s)

Clients

Description of the contract