Object-oriented programming:  
Role-based design

Topics:
- Collaborations with abstract roles
- Synthesis of multiple collaborations
- A "model" of reuse in role-based designs

Terminology

Collaboration: pattern of message exchange among a collection of objects to achieve some goal

Role: that subset of an object’s characteristics needed to fulfil its responsibilities in a collaboration

Observe: Both definitions refer to concrete objects; no mention of classes or interfaces

We would like to design reusable collaborations

Key is to model collaborations with abstract roles
Collaboration with abstract roles

```
GUI : Button
listener : ButtonListener

userEvent() --> buttonPressed(....)
```

Question

Consider the collaboration between the document manager and a viewport that displays its contents. Are there any opportunities for role abstraction in this collaboration?
**ViewPort: update collaboration**

```
(client)  vp : ViewPort  model : ViewPortModel
```

update()  retrieve(n)*

---

**ScrollBar collaboration**

```
GUI : ScrollBar  listener : ValuatorListener
```

userEvent()  announceNewValue(n)

---

CSE 370: Software Engineering  
K. Sörevågh
Exercise

Integrate a scrollbar object into our viewport--fileManager collaboration. Draw a sequence diagram to illustrate what happens when the user drags the slider handle to a new position.
Exercise

Synthesize concrete classes from which we could instantiate objects that play the roles in the new-position—update collaboration.

Exercise

Suppose we now want to add a reset button to the assembly. Pressing the button should clear the file. Obviously, the viewport should reflect that the file is empty. Draw a sequence diagram that illustrates this collaboration.
Synthesis

Defn: The process of constructing an application by composing multiple collaborations

Process:
- Gather together all relevant collaborations
- Identify roles to be played by the same object
- For each object, create an *adaptor class* to synthesize the role classes in the group for that object
- Write *configuration code* that allocates and links the objects

Synthesis: (1) align collaborations

![Diagram showing the alignment of ScrollBar, ViewPort, SubsequenceMap, and DocumentManager]
Synthesis: (2) compose roles

Synthesis: (3) design adaptor classes

No need to adapt Button and View
- Only role classes in their columns.

Other columns contain multiple role classes:

```java
class VpModel : public ValuatorListener,
    public AdjustableSequence,
    public SubsequenceMap
{ ... };

class MyDocumentManager : public Sequence,
    public DocumentManager
{ ... };
```
Synthesis: (4) model configuration

Synthesis (5): Write config. code

```java
//
Viewport vp;
ScrollBar sb;
VpModel vpm;
MyDocumentManager mMyMgr;

sb.registerListener(&vpm);
vp.registerModel(&vpm);
vpm.registerView(&vp);
vpm.registerModel(&mMyMgr);

Note: Bidirectional dependency requires both collaborators to point to one another
```
Virtues of role-based designs

Reuse: With a well-designed library, much of the work in building an application is concerned with the synthesis of collaborations
- Good, because synthesis is relatively mechanical
- Requires collaborations to be small and abstract

Program understanding:
- Easier to understand roles within one collaboration than objects that are playing roles in many collaborations
- Documentation should include sequence diagrams; i.e., can’t rely only on comments in the header files for role and adaptor classes.

Explanation

We composed classes from two different hierarchies to form a class for use in one application

Hierarchy

```
Button

<!interface>>

ButtonListener
```

Hierarchy

```
DocManager
```

CSE 370: Software Engineering
K. Sörensen
Inheritance to compose application-specific classes

Common in *mature* domains:
- Applications achieve reuse by extending and integrating classes from one or more hierarchies
- Class libraries designed in this fashion are extremely valuable corporate assets

Related to the idea of *program families*

*"We consider a set of programs to constitute a family, whenever it is worthwhile to study programs from this set first by studying their common properties and then determining the special properties of the individual members."*

*D.L. Parnas [TSE'1976]*

What is a “Family”?

Is Netscape a “program”?

Netscape 1.1 → Netscape 1.2 → Navigator 2.0 → Communicator 2.0 → Mosaic → Netscape 6.0

*All of these are internet browsers that do about the same thing = family*
How Are Families Developed?

Sequential-completion model of program evolution

from D. Parnas

Complete

Intermediate stages are refined to the next intermediate stage

These contain deferred implementations and/or deferred design decisions.

Abstract decisions model of families
Families and OOP

Lots of support for program families in OOP
  – Collaborations are like an abstract program, in which
certain design decisions have been deferred.
  – Class hierarchies correspond to *reusable* sets of
design decisions that are appropriate for all
members in a program family
    • Important, because there will often be many possible ways
to structure your class hierarchies.