purposes

after this class you’ll understand

why data models are useful
what a data model is and what it means
how to construct a data model

key ideas
separation of concerns: rep vs content
abstract structure with relations & sets
data models as invariants
what is data modeling for?
classic domain modeling

example: MIT WebSIS
information system for registration etc

before writing code, need a domain model
semesters, classes, students, etc

need answers to questions like
can a class last more than one semester?
can a student register for overlapping classes?
can a student register for multiple semesters?
modeling new structures

example: RunPee
“because movie theaters don’t have pause buttons”

questions to guide design of data structures
is start time and duration sufficient?
rank segments by criticality to story?
label with scene type (violence, sex, conversation)?
exercise: shortener
URL shortener challenge

imagine building a URL shortener
permanent shortcuts owned by users
expiring shortcuts not owned

design the data content
what data is stored, not how
any notation you like: diagram or text

goals
succinct, precise, expressive

www.yellkey.com/bad
URL shortener: review

does your model say if you can have...
  two users for one short?
  two shorts for one URL?
  permanent shortcut with no owner?
  a short that is expiring *and* permanent?
  a short’s target being changed?

does your model include coding details?
  generated keys or ids
  rep choices like arrays

how easy to modify or extend?
  make expiring shorts owned too
  have shorts owned by accounts with multiple users
why data models?
separation of concerns

what the data is  vs  how to represent it

a tool for thinking
one concern at a time: a way to make progress
helps expand representation possibilities
surprise: focus exposes subtle issues in what data is

other advantages
can separate roles: designer vs. implementer

biggest idea in programming?
But nothing is gained — on the contrary! — by tackling these various aspects simultaneously. It is what I sometimes have called “the separation of concerns”, which, even if not perfectly possible, is yet the only available technique for effective ordering of one’s thoughts that I know of. This is what I mean by “focussing one’s attention upon some aspect”: it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect’s point of view, the other is irrelevant. It is being one- and multiple-track minded simultaneously.

Dijkstra, On the role of scientific thought (EWD447)
data is central

state characterizes behavior
invariant is like a planet’s orbit

not just databases
rich transient state too
wanted...

a notation for data
lightweight, minimal & easy to learn
more succinct than code
more precise than sketches

representation independent
leaves rep choices open
not tied to a particular database
syntax
example email app
IT WILL DO YOU GOOD.
Classify

Spread recycling!! To save limited natural resources for our children’s future.

© 1991 Produced by Super Planning Company Limited
identify sets of atoms

Message

a SET of messages

Address

Folder
specialize sets

- Italicized subsets are exhaustive
- A folder is user-defined or predefined
- Classification is IMMUTABLE: Inbox can’t become Trash
- These happen to be singleton sets (but might not be)
identify dynamic subsets

- **Message**
  - not italicized: may have messages that are not unread

- **Unread**
  - no immutability marking: a message can be unread now and not unread later
relate atoms (properties)

Message → Address

relation is target
IMMUTABLE: to address doesn’t change
relate atoms (structure)

- don’t want one relation represented by two arcs

- a generalization, introduced to unify relations
determine multiplicities

Message ! to Address
from 

Object

Folder ? Message

contents

exactly one

one or more

at most one
putting it all together

Diagram:
- **Object**
  - **Folder**
    - **UserDefined**
    - **Predefined**
  - **Message**
    - **Unread**
      - **Inbox**
      - **Sent**
      - **Trash**
      - **Address**
      - **from**
      - **to**?
adding textual constraints

one inbox, trash and sent folder
predefined folders not contained by or contain others
no unread messages in sent folder
contents is acyclic
syntax summary sets

\[ A \subseteq B \]

\[ A \subseteq B \]

once an A, always an A

\[ A_1 \subseteq B \land A_2 \subseteq B \]

\[ A_1 \cap A_2 = \emptyset \]

\[ B \subseteq A_1 \cup A_2 \]
syntax summary relations

R ⊆ A × B

- over time, each A is mapped by R to the same Bs
- R maps each A to n Bs
- R maps m As to each B
- + one or more
- * zero or more
- ! exactly one
- ? at most one
- omitted = *

short for:

used when R1 and R2 share same properties
exercise: shortener revisited
construct a data model

**features**
- permanent and expiring shorts
- permanent shorts owned by accounts
- users can share accounts
sample solution
semantics
model = set of instances

\{ (x, y) \mid x, y \in \mathbb{N} \land x+y = 4 \}\)

syntax: a constraint on a pair of numbers

\{ (0,4), (1,3), (2,2), (3,1), (4,0) \}\)

semantics: a set of instances

syntax: model

semantics: set of instances
example satisfying instance

\[ \epsilon \in \{\text{Inbox}, \text{Sent}, \text{Trash}\} \]

\( \epsilon \) is one Inbox and one Trash and one Sent

no Predefined\.contents & Folder

no Sent\.contents & Unread

acyclic [contents]
example relating wrong kinds

one Inbox and one Trash and one Sent
no Predefined.contents & Folder
no Sent.contents & Unread
acyclic [contents]
example multiplicity violation

one Inbox and one Trash and one Sent
no Predefined.contents & Folder
no Sent.contents & Unread
acyclic [contents]
example multiplicity violation

one Inbox and one Trash and one Sent
no Predefined.contents & Folder
no Sent.contents & Unread
acyclic [contents]
example textual constraint violation

\[ \text{one Inbox and one Trash and one Sent} \]
\[ \text{no Predefined.contents & Folder} \]
\[ \text{no Sent.contents & Unread} \]
\[ \text{acyclic [contents]} \]
monotonicity

adding diagram features adds constraints, so you can grow the diagram

from \subseteq \text{Message} \times \text{Address}

all \text{m}: \text{Message} \mid #\text{m}.\text{from} = 1

from \subseteq \text{Message} \times \text{Address}

all \text{m}: \text{Message} \mid \text{m}.\text{from} \text{ unchanging}