OO Design Notations

Original version by B.Rogers/M.Utting

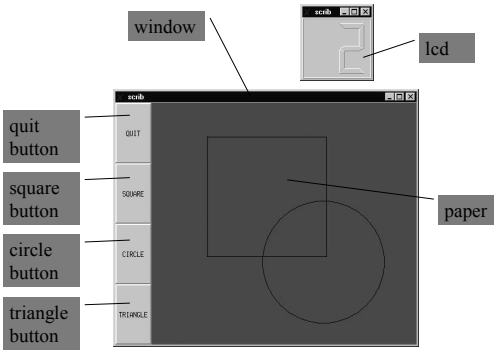
Notations

- There are a large number of OO and other design notations (text or diagram)
- Serve two main purposes
  - helping express ideas during design
  - documenting the final structure of a system
- Informal (personal)
  - allow free expression of ideas
- Formal (standard)
  - may admit proof of correctness
  - permits development of tools (Rat Rose)
  - communication to others

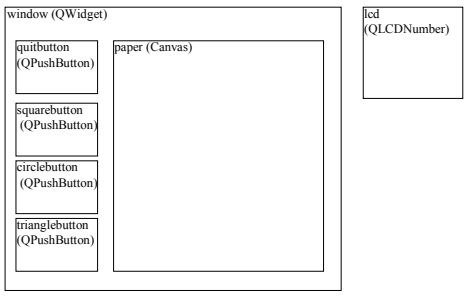
Notations

- Ad Hoc notations
  - widget relationships
- UML
  - class diagrams
  - (activity diagrams)
- CRC cards
  - Class, Responsibility and Collaboration
- Use Cases

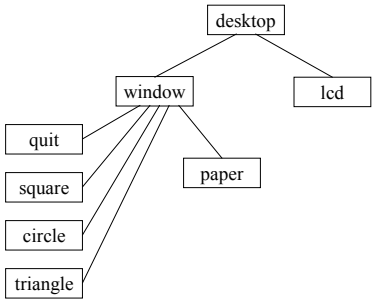
Widget Organisation



Organisation of Widgets



Widgets emphasising links

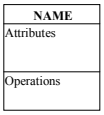


Unified Modelling Language

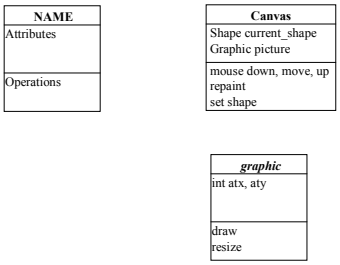
- Facetiously called Union of all MLs
- (UML) Grady Booch, Jim Rumbaugh, and Ivar Jacobson
- Unifies and standardises a number of notations developed over the 80 and 90's
- A (mainly) graphical language used to express object oriented designs.
  - use case diagrams; class diagrams; interaction diagrams; package diagrams; state, collaboration, sequence, activity diagrams; deployment diagrams.

Class Diagram

- Can be used at different stages of development: from modeling the kinds of objects in a system through to detailed implementation
- Classes may be implemented as Java classes or not, may just be kinds of items in a system
- A class is shown as a box:



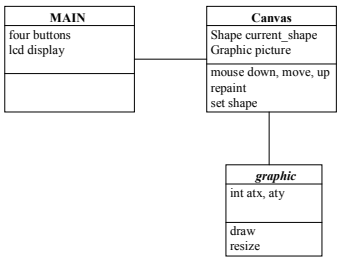
Examples

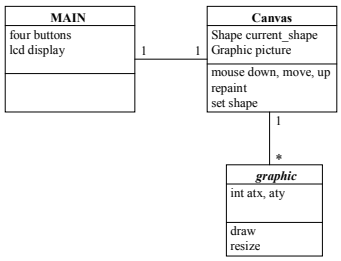


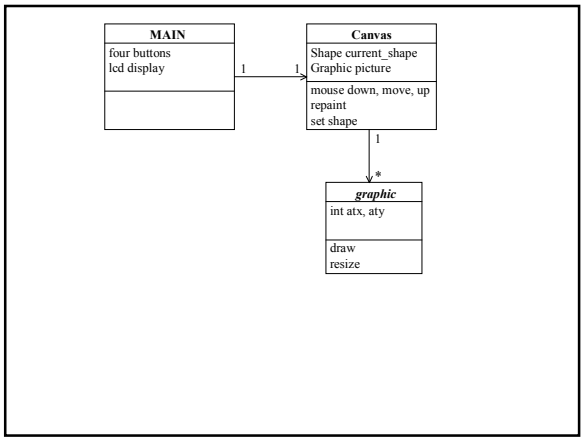
- Name in italics for an abstract class

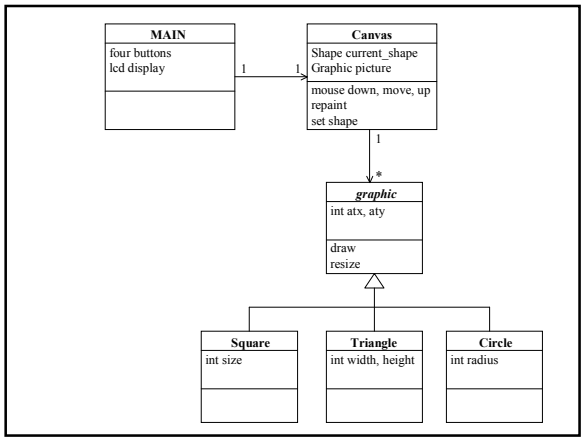
Class Diagram

- Association
  - what uses, has, contains, is related to ...
- Multiplicity
  - how many of each kind of object are related
- Navigability
  - what pointers are present
- Generalization
  - sub/super class relationships
- Other constraints



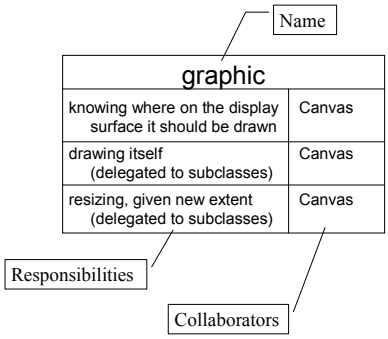






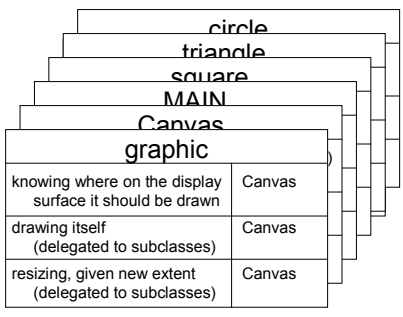
CRC Cards

- Class, Responsibility and Collaboration
  - Cunningham and Beck from Tektronix labs in Portland, Oregon. (Smalltalk)
  - Used 15 by 10cm index cards
- Attempt to encourage a high level description of the purpose of a class
- Avoid details of methods and attributes
- Small card limits the number of responsibilities and amount of detail



graphic	
knowing where on the display surface it should be drawn	Canvas
drawing itself (delegated to subclasses)	Canvas
resizing, given new extent (delegated to subclasses)	Canvas

Canvas	
responding to mouse events by drawing a graphic	Graphic(+)
knowing which kind of graphic to draw next	MAIN (buttons)
maintaining a list of graphics which have been drawn	Graphic
repainting the screen	Canvas



Use cases 1

- What is a use case?
- Identify user goal's, not system functions
- E.g. for a word processor:
  - Build an index
  - Insert a picture
  - Ensure consistent formatting.
- Describe external view, interactions with the outside world.

Use cases 2

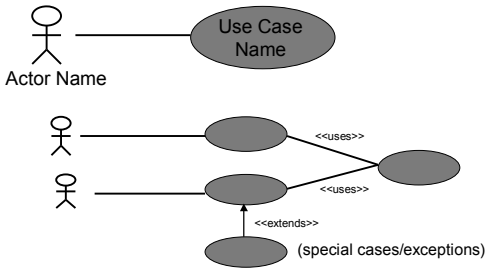
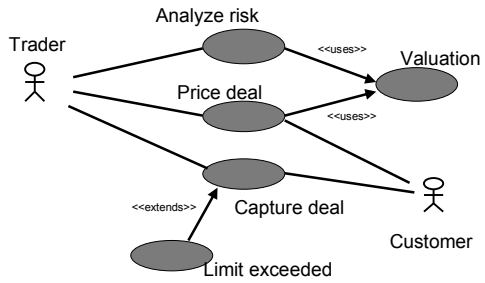
Express *requirements* at a high level

- The first step in software analysis is to develop a sufficient set of use cases. (but will also useful later on for system testing)
- Each use case (business task) has:
  - A name:
  - An *actor*: a role that a user takes when using the system. Actors are usually human.
  - A description: of interaction with system.

Description should tell us:

- What are the main tasks done by the actor?
- What inputs/outputs will the actor give/get?
- Does the actor have to inform the system about changes in the external environment?
- What information does the actor want from the system?
- Does the actor wish to be informed about unexpected changes?

Use Case Diagrams



Use Case Diagrams 2