The design and implementation of a notional machine for teaching introductory programming

Programming is hard

- Various studies show high drop-out rates for such courses
- Why is this?
  - One hypothesis is that students don't understand the "properties" of their program and how they are controlling them via code.

What is a Notional Machine?

- Originally proposed by Boulay (1986)
- Theoretical abstraction designed to represent how a particular program executes.
- Can provide one, or several metaphorical "layers" on top of the real machine
- Doesn't have to represent everything in the real machine
  - But must be consistent, be able to explain observed behaviour within its model.
  - Doesn't have to explain all characteristics of the real machine, but if not must have a well-defined subset.

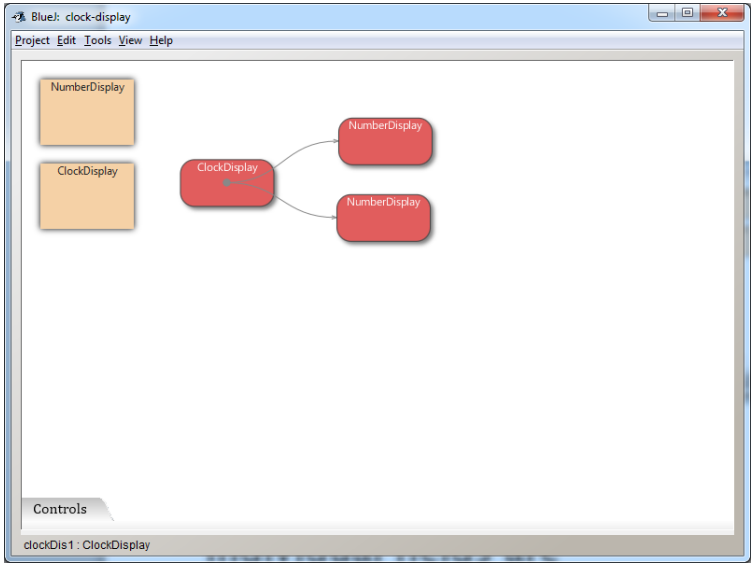
Aims of this work

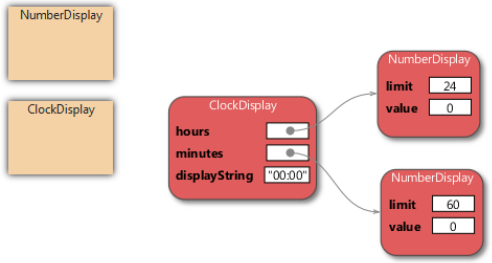
- To provide a notional machine that's useful for the teaching of introductory programming
  - Throughout the first year
- To provide a valid mental model for learning and reasoning about OOP
- To provide a common framework that teachers and students can refer to when describing OOP programs
  - Whiteboards, textbooks, software…

Demo

Basic representation

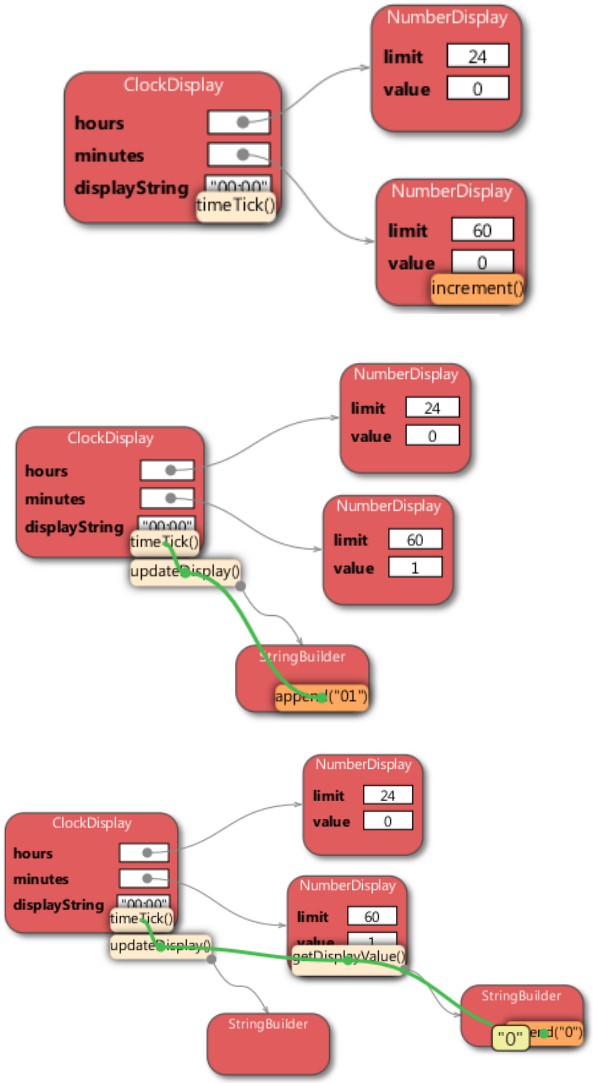
- Similar to “Objects First” textbook
- Classes are peach rectangles, objects are dark red rectangles with rounded corners
- Objects can be expanded, then individual fields are shown





Representation of execution

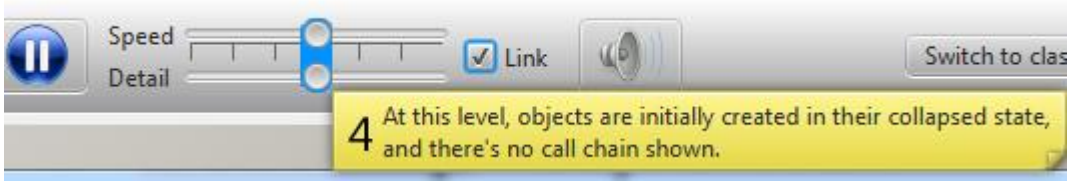
- Methods are displayed as orange rectangles; top most method is highlighted
- The “call chain” (stack trace) is displayed as an arrow overlaying the method calls
- Parameters are shown being passed along the call chain in “boxes”



Conceptual levels

- Some other similar tools show high levels of detail, visualising each atomic operation
  - Great (arguably) to start with, gets tedious quickly
- As execution speed increases (user controlled) less detail is shown
- At the slowest level, everything is shown – objects are expanded
- At the fastest level, only objects are shown in a “heatmap” style view

Conceptual Levels (2)



- At present, this is implemented with two sliders which can be “linked” together
- 7 different user-controlled “conceptual levels” at present

Summary

- The notation provides a diagram that can be consistently used in a number of formats
- The implementation animates a diagram from a live running program
- No separate stack trace view – heap and stack merged into one diagram
- Two separate cases for this work – the understanding of programming constructs, and the understanding of a program.

Future work

- Layout – layout in the prototype is sporadic and arbitrary at present, objects should be laid out more consistently
- Testing – should test the prototype with students, gain feedback and undergo multiple iterations of improvement