More on Design by Abstraction

CS151

Chris Pollett

Oct. 24, 2005.

Outline

- Animation Sort Algorithm Case study
- Applications Frameworks

Animation Sort Algorithm

- Book gives a sequence of programs which illustrate design by abstraction.
- The goal was to produce a program which could be used to animate the sorting of a list in an applet.
- Initially, everything is done in one class abstract AlgorithmAnimator which is subclassed from DBAnimationApplet.
- This class defines three methods:
  - abstract algorithm() - which controls how to sort,
  - run() - which is called from a template to get continue sorting
  - pause() - called inside algorithm to allow for an update in painting
- From DBAninmationApplet one still has paintFrame is also abstract.

Class Sort

- This class implements AlgorithmAnimator and has the following methods:
  - scramble() - creates an array of integers of size based on the display area. Then randomly swaps entries.
  - paintFrame()
  - bubbleSort()
  - quickSort()
  - initAnimator() -- uses applet arg parameter to say which sort algortihm to use
  - algorithm()
  - swap()

Separating algorithm()

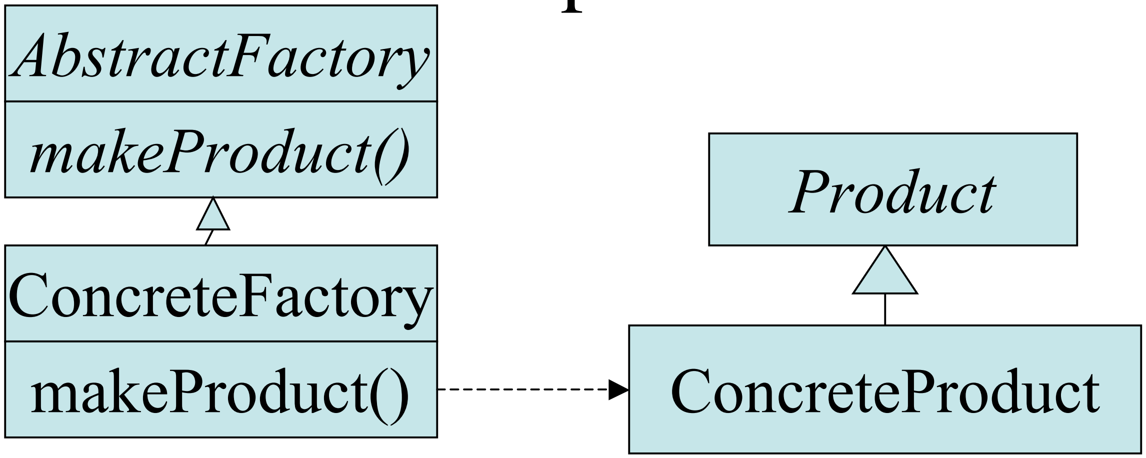
- The sorting algorithm is only loosely related to the actual animation process.
- So it is a more flexible design if we separate the sort algorithm out of the applet into its our class.
- The idea is that if a class contains components that address different concerns, then these different concerns are candidates to be split into their own classes.
- So the second version of the sort animator, Sort2, still extends AlgorithmAnimator, but has on it a SortAlgorithm object whose sort() method is invoked to do the sorting.
- Different subclasses of SortAlgorithm are created for different sort algorithms like bubblesort etc.
- The parameter tag of the applet now used to say which of these SortAlgorithms to instantiate

Improving initAnimator

- Rather than have the job of figuring out which SortAlgorithm to call inside initAnimator, it is better to split this out into a separate *factory* class, AlgorithmFactory.
- AlgorithmFactory has a method makeSortAlgorithm(String algName);
- The advantage of this is that now any time a new SortAlgorithm is created there is one place where we can create new objects of type sort algorithm based on a string name.

Factory Pattern

- Category: creational
- Intent: To define an interface for creating objects but let subclasses decide which class to instantiate and how
- Applicability: The factory design pattern should be used when a system should be independent of how its products are created.



Separating Display Strategies

- The way we actually do the display step can also be separated out of the Sort2 AlgorithmAnimator.
- To do this we can subclass Sort2 to make Sort3 which has a protected SortDisplay object which is called when paintFrame is executed.
- A SortDisplay is just an interface with a method display to do the drawing.
- The book give four implementations of this HSortDisplay, VSortDisplay, etc,
- One can also make a factory for SortDisplay’s

Application Frameworks

- An *object-oriented application framework* or just *framework* is a set of co-operating classes that represent reusable designs of software systems in a particular application domain.
- For example, the Java Collection Framework, the AWT and Swing, or the Input/Output Framework.
- Some characteristics of frameworks are:
  - Extensibility
  - Inversion of control
  - Design Patterns as Building blocks.

Design Requirements for Frameworks

- Completeness
- Adaptability
- Efficiency
- Safety
- Simplicity
- Extensibility

The Collections Framework

- A *collection* is an object that contains other objects.
- The Collections framework has abstract collections for four major categories of collections:
  - Bags: (aka multisets) unordered collections which allow repeats. Ex {1, 2, 3,2, 4}
  - Sets: unordered collections no repeats. Ex {1, 2, 3, 4}
  - Lists: ordered bags (1, 2, 3, 2, 4) would be different from (1, 2, 2, 3, 4)
  - Map: a set of key, value pairs. {(k_1, v_1), (k_2, v_2), …} such that each k_i is distinct.