More on Design by Abstraction

CS151

Chris Pollett

Oct. 19, 2005.

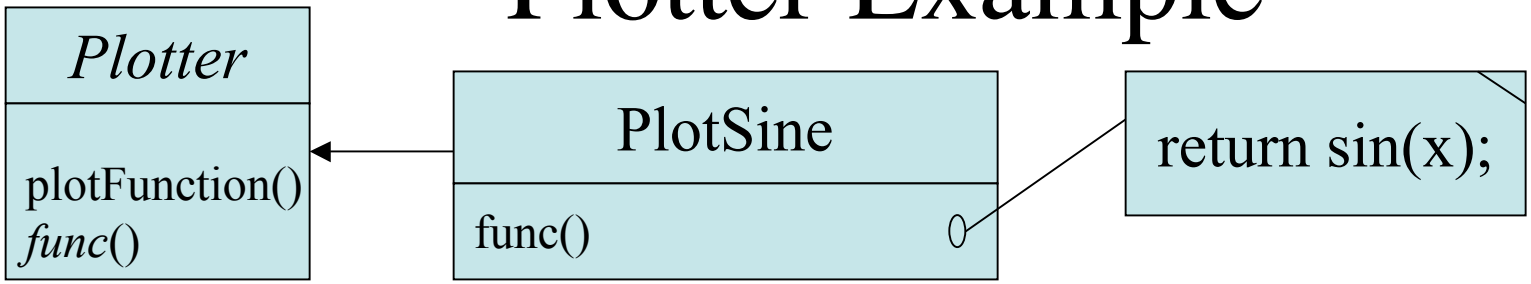
Outline

- Generalizing
- Design Pattern -- Strategy
- Abstract Coupling
- Design Pattern -- Iterator
- Case Study

Generalizing

- **Generalizing** is a process that takes a solution to a specific problem and restructures it so that it solves not only the original problem but a category of other problems as well.

Plotter Example



- The book gives a generic function plotter program to illustrates the generalization technique as well as the Strategy Pattern.
- The original function plotter class shown is Plotter. Plotter uses a template pattern with hook function func() to say what to plot. PlotSine is a concrete subclass of Plotter
- Plotter is generalized to MultiPlotter which can plot multiple single variable functions overlaid on the same 2D-space.
- The problem that needed to be solved was how to separate the functions to be plotted from the plotter.

One way to solve Plotter Problem

- Could add more hook methods to the method:

```
protected void plotFunction(Graphics g)
{
    for(int px=0; px.dim.width; px++)
    {
      try{        double x =
                              (double)(px-xorigin)/(double)xratio;
                  double y =func1(x); //get ready to plot first function
                  int py = yorigin - (int)(y*yratio);
                  g.fillOval(px-1, py-1, 3, 3); //plot first function
                  y =func2(x); //get ready to plot second function
                  int py = yorigin - (int)(y*yratio);
                  g.fillOval(px-1, py-1, 3, 3); //plot second function
      //…
      }catch(Exception e){}
    }
}
```

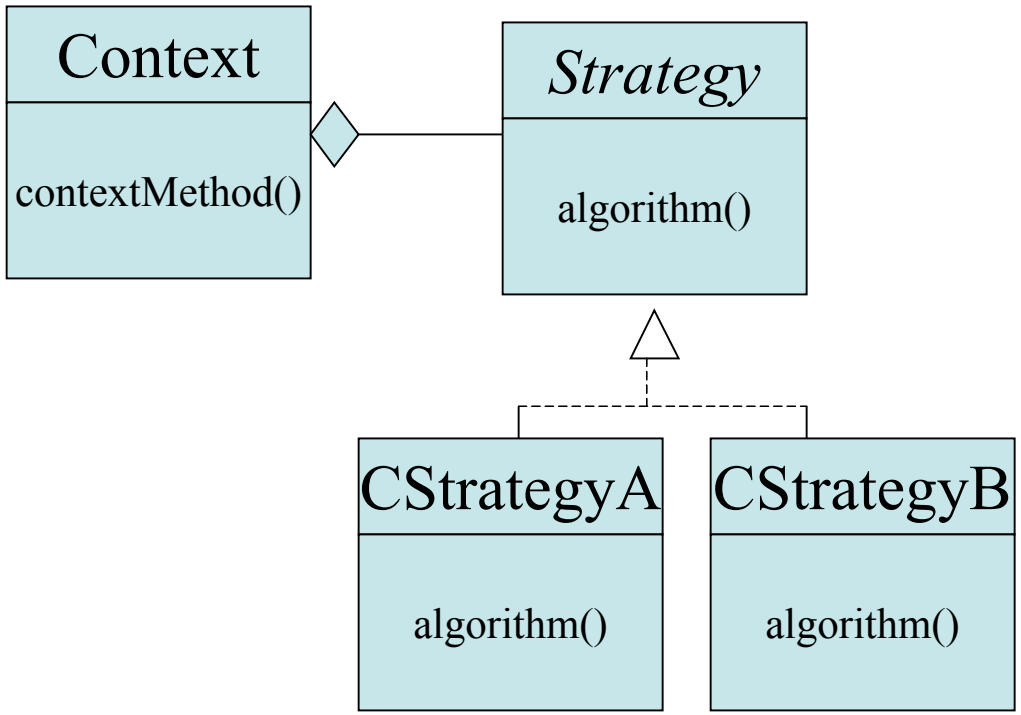
- This would only allow one to plot a fixed number of functions.

More on Plotter Problem

- Instead, we create an interface Function which has a method apply(double x) to compute f(x)=y.
- This interface can be implemented to return sin x, cosine x,etc.
- Then MultiPlotter is made an abstract class with an abstract initMultiPlotter() method.
- Implementations of this spell out which functions we will plot.
- MultiPlotter has a LinkedList of Function objects, and an addFunction method to add to this list (for instance in init).
- To do plotFunction we now cycle over the functions in this list plotting each one in turn.

Design Pattern -- Strategy

- MultiPlotter illustrates the Strategy Pattern (another example is LayoutManager in the AWT)



In MultiPlotter scenario:

• Strategy is Function

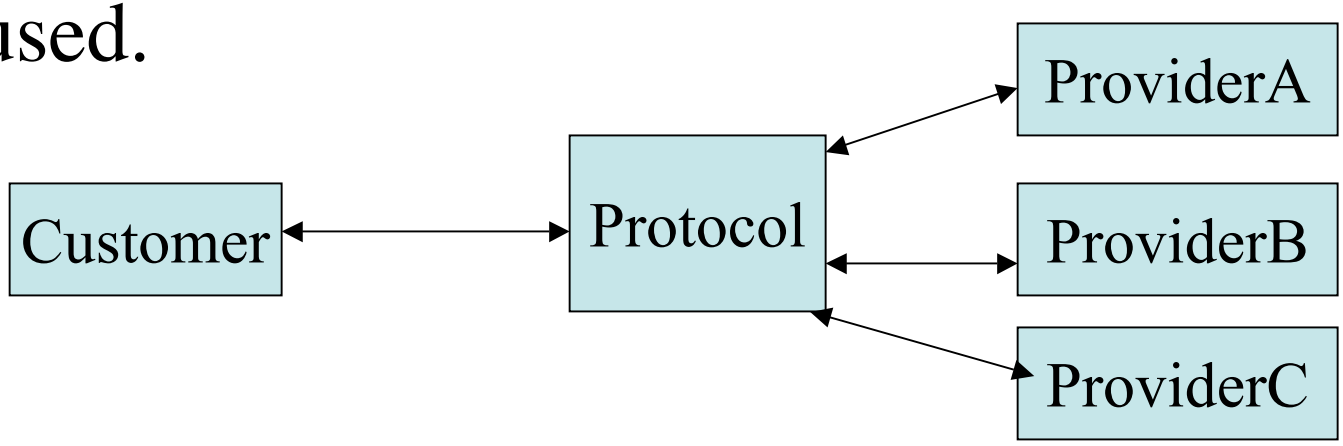
•CStrategy's would be classed like Sine or Cosine

•Context would be MultiPlotter which maintains a reference to one or more Strategy objects (in this case Function objects).

•contextMethod in this case would be plotFunction which cycles over the Function objects and plots them in turn.

Abstract Coupling

- **Abstract Coupling** refers to how clients couple with service providers: A client accesses a service provider through an interface without knowing which concrete implementation of the interface is being used.



Design Pattern -- Iterator

- An example of abstract coupling is the use of iterators.
- The idea is we have many different kinds of container class and we'd like to be able to cycle over the elements from any different one.
- We create an interface Iterator and have each container class have a method iterator() which returns an object of this type.
- An Iterator supports the methods reset(), next(), and hasNext() to allow us to cycle over the element of the class.
- Enumeration's in Java are an older related idea which support just hasMoreElements and nextElement();