designing a stock quoter

Daniel Jackson
February 13, 2008
announcements

MEET announcement
\* Max Goldman

about lecture exercises
\* do soon after lecture
\* should not be a big burden
\* some questions quite open

today
\* one handout (code)

to do
\* LNBs due tomorrow
\* project & lab 1 out today
\* read lab by Friday

about feedback
\* please do immediately; should only take 3 minutes

about Visio
\* need CSAIL certificate currently
\* will have on Stellar soon, only MIT certificate needed
plan for today

comment

› ignoring events in State pattern

designing a stock quoter

› plugins for different formatting
› using Java interfaces

mid-lecture break

› your first project: a multipart downloader
› Mitchell and Webb on downloading
designing a stock quoter
quote generation problem

problem

- want to obtain stock quotes for some ticker symbols
- produce both RTF and HTML output
- put ask price in bold if change since open $\geq \pm 1\%$
design tasks

tasks, for each ticker:

› download quote information from website
› parse to extract stock quotes
› write to file in RTF or HTML format

what’s likely to be painful?

› parsing, so minimize by choosing a site with simple format
› choose Yahoo: offers quotes in comma-separated-values ("CSV")

task example

› http://quote.yahoo.com/d/quotes.csv?s=aapl&f=oap2
› returns the string 130.75,125.20,"-3.55%"
public class Quoter {
    private URL url;
    private String open, ask;
    private int change;

    public Quoter (String symbol) throws MalformedURLException {
        url = new URL("http://quote.yahoo.com/d/quotes.csv?s="+symbol+"&f=oap2");
    }

    public String getOpen () {return open;}
    public String getAsk () {return ask;}
    public int getChange () {return change;}

    public void obtainQuote () throws IOException {
        BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream()));
        String csv = in.readLine();
        in.close();
        StringTokenizer tokenizer = new StringTokenizer(csv, "");
        open = tokenizer.nextToken();
        ask = tokenizer.nextToken();
        change = (int) (100 * (Float.valueOf(ask)-Float.valueOf(open))) / Float.valueOf(open);
    }
}
questions

why are the fields of Quoter private?
  · what might happen if they weren’t?

Quoter is a state machine
  · what pattern is being used?
  · draw the state machine

BufferedReader is a state machine
  · what pattern is being used? (guess)
  · draw the state machine
design options

decoupling achieved so far
• which website, format, etc from stock quote

design option #1
• just build two formatters that use Quoter

design option #2
• formatter takes flag (HTML or RTF)
• tests flag to determine flow of control

a principle
• localize each design decision in exactly one place
• more crudely: “don’t repeat yourself”
a better solution

factor out

• generator: decides *how* to put in bold, italics, etc
• formatter: decides *what* to put in bold, italics, etc

designing the generator

• make it a state machine!
• two versions, one RTF and one HTML
• but *same interface*
generator machine

key design idea

- develop generic interface for text formatting
public class RTFGenerator {
    boolean italic, bold;
    String filename;
    PrintStream stream;

    public RTFGenerator (String filename) {
        this.filename = filename;
    }

    public void open() throws FileNotFoundException {
        FileOutputStream fos = new FileOutputStream (filename);
        stream = new PrintStream(fos);
        stream.println ("{\rtf1\mac");
    }

    public void close() {
        stream.println ("}"");
        stream.close();
    }

    public void newLine () {
        stream.println ("\n");
    }

    public void toggleBold() {
        stream.println (bold ? "\\f\\b0" : "\\f\\b");
        bold = !bold;
    }

    ...
}
the big question

how to make formatter independent of generator?

- want them decoupled
- can plugin different generator
- without changing code of formatter

solution

- formatter doesn’t refer to particular generator
- refers to interface instead
interfaces, in pictures

what we want

• two ways to configure formatter

how does formatter refer to generators?

• with an interface

formatter

RTF generator

HTML generator

formatter

uses

generator

RTF generator

HTML generator

generator

implements
public class RTFGenerator implements Generator {
    boolean italic, bold;
    String filename;
    PrintStream stream;

    public RTFGenerator (String filename) {
        this.filename = filename;
    }

    public void open() throws FileNotFoundException {
        FileOutputStream fos = new FileOutputStream (filename);
        stream = new PrintStream(fos);
        stream.println ("{\rtf1\mac");
    }

    public void close() {
        stream.println ("}");
        stream.close();
    }

    public void newLine () {
        stream.println ("");
    }

    public void toggleBold() {
        stream.println (bold ? "\f\b0" : "\f\b");
        bold = !bold;
    }
    ...
}
using generator

public class QuoteFormatter {
    Set<String> symbols = new HashSet<String> ();
    Generator gen;

    public QuoteFormatter (Generator gen) {this.gen = gen;}
    public void addSymbol (String symbol) {symbols.add (symbol);}

    public void generateOutput () throws Exception {
        gen.open ();
        for (String symbol: symbols) {
            Quoter q = new Quoter (symbol);
            q.obtainQuote();
            gen.write (symbol + " : ");
            gen.toggleItalic (); gen.write ("opened at "); gen.toggleItalic ();
            gen.write (q.getOpen ());
            gen.toggleItalic ();
            gen.write (" and is currently trading at "); gen.toggleItalic ();
            boolean bigChange = Math.abs (q.getChange()) >= 1;
            if (bigChange) gen.toggleBold();
            gen.write (q.getAsk ());
            if (bigChange) gen.toggleBold();
            gen.newLine();
        }
        gen.close();
    }
}
the crucial part

interface declaration

- method declarations
- no fields or constructors

```java
public interface Generator {
    public void open () throws Exception;
    public void close ();
    public void newLine ();
    public void toggleBold ();
    public void toggleItalic ();
    public void write (String s);
}
```
public class QuoteApp {

  public static void main(String[] args) throws Exception {

    Generator rtfg = new RTFGenerator("myQuotes.rtf");
    QuoteFormatter formatter = new QuoteDisplayer(rtfg);
    formatter.addSymbol("AAPL");
    formatter.addSymbol("INTC");
    formatter.addSymbol("JAVA");
    formatter.addSymbol("MSFT");
    formatter.generateOutput();

    Generator htmlg = new HTMLGenerator("myQuotes.html");
    formatter = new QuoteFormatter(htmlg);
    formatter.addSymbol("AAPL");
    formatter.addSymbol("INTC");
    formatter.addSymbol("JAVA");
    formatter.addSymbol("MSFT");
    formatter.generateOutput();
  }
}

plugin is selected here
what we’ve achieved

dependency diagram

- arc means “depends on”

just invokes QD with tickers & selects output format

obtains and outputs quotes

QuoteApp

QuoteFormatter

HTMLGenerator

RTFGenerator

Obtains quotes

Quoter

Generator

formats text in HTML

formats text in RTF
exercise

which modules would you need to modify to ...

‣ handle new RTF syntax for italics?
‣ put ask price in bold if down since open?
‣ use google finance instead of yahoo?
‣ add year-to-date change to report?
interfaces more generally
prototype

suppose you want

• client \texttt{C} to use plugins \texttt{P1} and \texttt{P2}

strategy

• design interface \texttt{PI} that captures commonalities of \texttt{P1} and \texttt{P2}
• \texttt{C} uses \texttt{PI}
• \texttt{P1} and \texttt{P2} implement \texttt{PI}

code

```java
class C {
    C (PI plugin) {...} \textcolor{blue}{client class uses interface}
    ...
    plugin.m ();
}

interface PI { T m(); } \textcolor{blue}{declare interface}

class P1 implements PI { T m () {...} } \textcolor{blue}{plugins implement interface}

... C c = new C (new P1()); ... \textcolor{blue}{pass plugin object to client}
```
how compiler helps

a guarantee

• every method called on an object exists
• this is called static typing

how static typing works for interfaces

• compiler checks
  (1) that client uses only methods in the interface
  (2) that each plugin has all methods declared in the interface
• so if client calls method m
  then interface has m (because check 1 passed)
  and plugin has m (because check 2 passed)
other uses of interfaces

decoupling from choice of representation

\• very common and important
  \- List$<$NoteEvent$>$ recording = new ArrayList$<$NoteEvent$>$ ();
  \- recording.add (event);
  ...

“marker” interfaces

\• declare no methods
\• used to expose spec properties (eg. java.util.RandomAccess)
\• or as hack to add functionality (eg. java.io.Serializable)
summary
what did we do?

a general strategy

• client should know only about **spec** of service it uses
• so **decouple** by **interposing** spec between client & service

language-specific issues

• how is spec declared? (Java: with an interface)
• how is service plugged in? (Java: constructor call, pass object)

dependency diagrams

• show essence of code design
• missing dependences are the interesting ones!
lecture exercise

by next lecture (Monday) in your LNBs

understanding stock quoter code

- [easy] Why are the open and ask prices represented as strings in Quoter, and not as integers? Is this a good design choice?
- [moderate] The design of RTFGenerator and HTMLGenerator is coupled to the decision to use file output. Sketch how to modify the design to support different kinds of output.
- [moderate] Suppose RTFGenerator and HTMLGenerator were implemented with the Singleton state machine pattern. What implications would this have for the design?

understanding Java collections

- [easy] Read the documentation for java.util.List and for its implementations. When would a client of a list want to refer to a particular implementation?