

CS1110 Lecture 09 28 Sept
Developing (String-processing) programs;
class Vector; wrapper classes

Have your iClickers out.

Reading for Thursday's lecture: pp. 403—408, skipping section 15.1.2
* Recursion can be a difficult topic, but we'll make it easy.

Prelim: 7:30-9PM Thursday 7 October.

Last name A-K: go to Olin Hall 155. Last name L-Z: go to Olin Hall 255.

If you have a conflict and *didn't* receive an acknowledgment email yesterday, email Maria Witlox, mwitlox@cs.cornell.edu TODAY.

- Past prelims are posted to the course website. Use DrJava to check *your* answers!
- Thursday: A handout will explain what is on prelim 1
- Sunday: Review session, 1-3PM, Phillips Hall 101 (if you miss it, the slides will be posted)
- A3 is due Wed night on the CMS. Form any groups beforehand.

1

An application: String processing, stepwise refinement,
usefulness of Javadoc, problem solving

Strings are a particularly important type, because lots of information (especially non-numerical data) is stored in Strings.

For example, many webpages can, for many intents and purposes, be considered to be Strings.

Application: “scraping” (extracting) live stock quotes from the Web:
`getQuote("goog")` will print out Google's [ticker symbol: "GOOG"] current stock price, and store a list of all previous stock-price requests;

`showRecord()` will return something like this:

```
"[aapl @ Mon Sep 27 10:00:40 EDT 2010: $294.05, aapl @ Mon Sep 27 10:00:48 EDT 2010: $293.7, goog @ Mon Sep 27 10:09:02 EDT 2010: $534.38]"
```

Reminder: Principles and strategies

Develop algorithm step by step, using principles and strategies embodied in “stepwise refinement” or “top-down programming”. READ Sec. 2.5 and Plive p. 2-5.

- **Take small steps.** Do a little at a time
- **Refine.** Replace an English statement (*what to do*) by a sequence of statements to do it (*how to do it*).
- **Refine.** Introduce a local variable —but only with a reason
- **Compile often**
- **Intersperse programming and testing**
- **Write method specifications** —before writing the bodies
- **Separate your concerns:** focus on one issue at a time

Note the similarities to outlining and writing an essay!

3

Outline for writing class StockQuote

1. What information do we need to store?

- what objects? what should be in the objects, vs. what should be static? What types should the variables be?
 - a) How do we implement a list? (answer: Vectors)

2. What methods do we need? (Specify them carefully, and stub them in!)

- b) How do we implement list-based methods?
- c) How do we actually get stock-quote data?
 - i. how can we access web pages?
 - ii. can we treat their contents as Strings, since we're good at that?
 - iii. how can we convert String prices to numbers? (answer: Wrapper classes)

4

Let's answer question one. Below, we've omitted "private" for brevity.

A. String symbol; // ticker symbol (case insensitive)
 Date time; // time the quote was taken;
 double price; // price of the stock when quote was recorded
 ListOfStockQuotes record; // list of all requested quotes
 public static void getQuote(String s);

B. Same as A, but getQuote(String s) is not static

C. String symbol; // ticker symbol (case insensitive)
 Date time; // time the quote was taken;
 double price; // price of the stock when quote was recorded
 ↗ static ListOfStockQuotes record; // list of all requested quotes
 public static void getQuote(String s);

D. Same as C, but getQuote(String s) is not static

E. None of the above

C's only diff from A

5

Class Vector – for maintaining lists of objects [more in lab]

In the interactions pane, you can try the following (Person.java and StockQuote.java need to be in the working directory and compiled):

```
import java.util.*;
import javax.swing.*;
Vector v= new Vector(); // v can store any object
v.add(new JFrame());
v.add(new Person("Smith", 1990, false));
v.get(1) // returns toString for "Smith" (indexing starts at 0)
v.toString() // contents of entire vector, using each object's toString()
```

// Important syntax: record can only store (names of) StockQuotes.
 Vector<StockQuote> record= new Vector<StockQuote>();

6

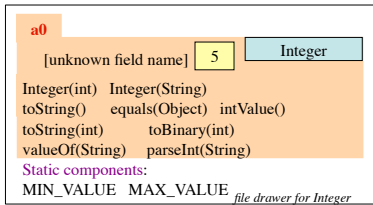
Wrapper classes – a way to treat primitive types as objects

- Sometimes objects are required; e.g., Vectors can only store objects:
 v.add(new Integer(5)); // Integer is an object version of int

[In newer versions of Java, v.add(5) is allowed; the non-object 5 is wrapped in an Integer object and the name of that object is added to v.]

- wrapper objects provide a place to store useful methods

An instance of class Integer contains, or "wraps", one (immutable) int value.



7

Each primitive type has a corresponding wrapper class.

Primitive type	Wrapper class	Each wrapper class has:
int	Integer	<ul style="list-style-type: none"> Instance methods, e.g. equals, constructors, toString, Useful static constants and methods.
long	Long	
float	Float	
double	Double	
char	Character	
boolean	Boolean	

Integer k= new Integer(63); int j= k.intValue();

You don't have to memorize the methods of the wrapper classes. But be aware of them and look them up when necessary. Use Gries/ Gries, Section 5.1, and ProgramLive, 5-1 and 5-2, as references.

8