Levi D. Smith CS2390 Modeling and Design

Class Schedule

Week 1

March 30: Class starts: Boggs B6, 12-1:30. First chapter of "Designing Object-Oriented Applications and their User Interfaces". Intro to Squeak. Chapters 1 and 17 of Goldberg and Robson

March 31: Lab 0 - optional, ungraded lab. See Labs

April 1: Chapter 1S: "Using Smalltalk". Chapters 2 and 3 of Goldberg and Robson.

Week 2

April 6: Finishing Chapter 1S.

April 7: Lab 1: Debugging in Smalltalk

April 8: Chapter 3: "Designing Objects".

Week 3

April 13: More on Chapter 3: "Designing Objects"

April 14: Lab 2: Using MuSwiki, CRCWindows, MAT annotations, and making classes

April 15: Live OOA/D/P. Go to the Alan Kay seminar on April 16 at 3 pm. Assignment 1 due in class

Week 4

Lab 3 starts Monday April 19 to noon April 21.

April 20: Finish Chapter 3: "Designing Objects". Some on Data structures in Smalltalk. Read it at http://www.cc.gatech.edu/classes/cs2390_97_summer/lectures/datastruct/slide1.html Chapters 9 and 10 of Goldberg and Robson.

Note: No class! We will instead meet for questions on irc.millenia.org port 6667 in the #gtcs2390 channel. Sessions will be 12-1 pm and 9-10 pm.

April 21: Lab 4 starts now and ends April 23 at 6 pm.

April 22: Review for midterm.

Note: No class! We will instead meet for questions on irc.millenia.org port 6667 in the #gtcs2390

channel. Sessions will be 12-1 pm and 9-10 pm.

Week 5

4/26 MIDTERM REVIEW - 6:30 CoC 101

April 27: MIDTERM

April 28: No Lab

April 29: Nuts and Bolts of GUIs in Squeak. Chapter 2S: "Joe the Box in Smalltalk". Chapters 18-20 of Goldberg and Robson. *April 20 is Drop Day*. Assignment 2 due.

Week 6

April 30: Grammars, Tokenizing, Parsing.

April 31: Lab 5 Blocks and Data Structures

May 6: Grammars, Tokenizing, Parsing in Squeak

Week 7

May 11: Chapter 4: "Designing UI Software". Intro to MVC

May 12: Lab 6 Fund with Drawing

May 13: User Interface Toolkits: MVC and Morphic. Assignment 3 due.

Week 8

May 18: Introduction to C++

May 19: Lab 7 Debugging in C++

May 20: Overview of Java. Choosing an O-O Language

Week 9

May 25: Designing User Interfaces

May 26: NO LAB

May 27: Introduction to Simulations. Assignment 4 due.

Week 10

June 1: WebServers and WebServers in Squeak. Mark's Research.

une 2: NO LAB

June 3: Review for Final.

Note: No class! We will instead meet for questions on irc.millenia.org port 6667 in the #gtcs2390 channel. Session will be 9-10 pm.

Back to the top | Over to CoWeb

CS 2390 March 30, 1999

http://www.cc.gatech.edu/classes/cs2390_99_spring

CS 2390 ⟶ CS 2340

we will not be using UML - large, few decent tools

no simulating

create an HTML parser

25% Midterm

33 Final 10% labs - lowest dropped

Design 1

2 Books - Goldberg - Purp / Yellow SMSO

- HTML book

1st Assignment Due April 15th

~~Allen~~ Alan Kay invented OO

15th lab 3 occurs before lab 3, and have to do it outside class.

lab 4, critique others designs

Assignment 1 - design server monitoring system

A2 - determine wether a page has changed

A3 - build a Web browser

A4 - build A1

Co-Web - Collaborative Website

~~OO History~~

Starting to program

structured programming

- Break down problem into separate functions
- Determine variables
  - Local Variables
  - Parameters - type, #
- Group Functions into classes

First OOs - Sketchpad + Simula

OO drawing

Simulate the real world started "class" + "multi tasking"

Alan Kay - Flex - first OO computer lang

- also came up with idea of PCs

Small Talk 1st OO lang

Small Talk - 72 - Overlapping Windows

Uses mouse, icons,

ST 80 - 1st ST released to the world

- platform independent - before Java

Squeak is original ST brought back to life in '95

'79 Bjame Stoustrup - C++

'96 Gosling - Oak - runs on everything (toasters)

later - Java

CS 2390 April 1, 1999

Concrete Stream
Standard FileStream

new Morph
200 Morphs - ~~[illegible]~~ In Image

Assignment #1 and Correction to Image Posted
SmallTalk Express

all computations are messages to object
messages
printString - many of these in ST
Every object is an instance of a class

3 local variables → | anArray anIndex aValue |

anArray := Array new: 10.
a = Array new: 12.

; ⇒ send the command to the previous object
all variables are pointers

block

b := | Transcript show: 'Hello'; cr. |.
b value.

Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello

b := 110 times Repeat: | Transcript show: 'Hello'; cr.
b value.

d at: 3 put: $R.

DWIM - Do what I need

later binding

<u>Key word Messages</u>
(-4) abs
90 sin
an Array at: 5
$a asUppercase

Parens

b := [: param | code here].

Sources
Image - memory of the virtual machine
aka "EXE file"

Need Image File and Executable
Only need sources and changes if you are doing actual development

Image - Sources & change

Image - Object code
change contains all of the codes that I've
executed - EVERYTHING IS HERE

April 6, 1999

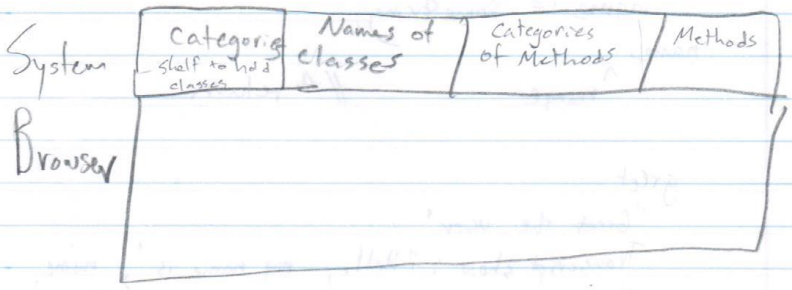
Sources File - Everyone has the same
Image File - Each person has a different one
Open Project - To begin work
Open - MVC project (yellow)
Click on it, then Enter
To go to top - previous project

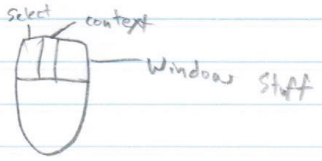
- Open WorkSpace - to execute commands
- Open Transcript -

Transcript show: (2 + 3) printString

need (2 + 3) → 2 + 3 printString

- Open Browser - System browser - lets you look at all of the source code in Squeak





Save your work
Click on background (desktop menu)
then - Save

Save AS - Fred.image and Fred.changes

Object subclass : #Muppet
instanceVariableNames : 'color name furzy'
classVariableNames : '' //global vars
(Alt-s) poolDictionaries : '' //ignores this
or category : 'My Category'
Accept

WS - | kermit |
kermit := muppet new.

Methods - name: someString
name := someString
- name
^name //^ return

greet
"Greet the user"
Transcript show : "Hell, my name is ", name.
Transcript cr.

Muppet Subclass: # FriendlyMuppet
Can't do super class from Workspace

FileOut → Saves as .st

X public private protected don't exist

Constructors in ST

new category → initalization

initalize
"Initialize my name and color"
name := 'faceless sock'.
color := 'dingy gray'.

class method
new
| newMuppet |
newMuppet := super new.
newMuppet initalize.
^newMuppet.

Muppet? yes

GM new → no

super is a variable

りばい・ど・すみす

リバイ・ド・スミス

Standard File Stream or Crlf Filestream
(Concrete filestream)
Read Chapter 1 of this book

Designing Object Oriented Programs
BOOST - Basic Object Oriented Support Tool
Case - do an additional analysis on your design

Designing Objects
Each object should only be responsible for itself

Analysis is language independent
Coding is language dependent

OO analysis

Objectory Method - big process
Standard UML - Unified Modeling Process

1.) What objects do I need?? Brainstorm them
Filter them - separate those of user interface from those which relate to the problem.
Are some of the nouns subclasses of the other ones.

Anal

Brain

CRC

Coed

Description

CRC Cards - OO Analysis

Start finding scenarios for the system.

Coad is a subset of UML

Clock

~~sec hand~~

~~hand~~

~~face~~

~~numbers~~

gears

springs

⋮

human interface

A4

newsgroups	Nouns
CoWebs	HTML
Webcrawl	Monitors
	URLs
	Checkers

Design it so I can reuse it in other places

Stop at when what you are creating doesn't add anything that isn't in any programming language

Delegating - calling another object to handle a method

SmT

Object subclass : #clock
instanceVariableNames : 'time timer displayFormat'

Smalltalk beep

process := | 100 times Repeat : | Transcripts show :
'TICK' ; cr ; || newProcess.
process priority ; (Delay forSeconds : 1) wait.
(Processor userBackgroundPriority).
process resume

Read Stream

Task for next lecture → Design an Alarm Clock

April 29, 1999

Box class methods for: 'initalization' stamp:
'mjg 3/27/98 21:55'!
new
^super new initalize !!

it will get initialized twice

Named Box class (Box Subclass)
new
^ super new initalize
① ④

Box class
new
^ super new initialize
② ③

---

namedBox new

initalize
position := 50@50,
size := 50,
tilt := 0.
(size)
clear World
Bitmap → (Form extent: 600@200) fill White display

>> Box clearWord.

Display is a kind of display

Creating Forms
f := Form from User
f display
f displayAt: 100@100
f edit "for drawing"
"Morphic Drawing"
"Reading
Form from FileNamed: 'my.gif'
or
gvu := HTTPSocket: http Gif: 'location
gvu inspect
gvu display.

Shrinking Forms

```
|f s|
f:= Form fromUser
s:= f shrink f boundingBox by: 2@5
    s display On: Display at: Sensor waitButton

s:= f magnify: f boundingBox by: 5@5
```

```
|a f|
f:= Form fromDisplay: (0@0: 200@200)
a:= "angle of rotation"
|Sensor anyButtonPressed| whileFalse:
    ((Form fromDisplay: (Sensor cursorPoint
    extent: 130@66))
    rotateBy: (a:= a+5) display|.
f display "fixes your display at the end"
```

BLT - binary / transfer

BitBlt

WarpBlt - color graphics - warp the result

```
|myPen|
myPen := Pen new.
myPen up.
myPen goto: position.
myPen turn: tilt.
myPen color: (Color black).
myPen down.
  4 times Repeat: [myPen go:
    size; turn 90].
```

(Sensor
global variable
Input Sensor

```
Sensor keyboard
Sensor keyboardPeek
```

```
NamedBox
  |name|

george := NamedBox Named: 'George'.
```

```
named: aName
    |newBox|
    newBox := Super new.
    newBox name: aName.
    newBox draw.
    ^ newBox.
```

tek - data for printer

you have to use 'super' to access the instance methods of Box (superclass)
self has the value of what was originally created
self will be box if we do:
new Box := Box new.
but if we want self to be named Box, then do:
new Box := Super new.

```
Box perform: #clearWorld
 -or-
message := #clearWorld.
Box perform: message.
-or-
receiver := Box.
message := #clearWord.
receiver perform: message.
```

means symbol → only one symbol per name.

CS 2390 May 4, 1999

Tokenizing / Parsing

Syntatic Analysis - how tokens fit together to work.
Finite Stat Au-tom-ata

3 steps - Lexical Analysis
Syntatic Analysis - figure out if tokens are in right order.
Backend: Create the object code for the right syntax

The code that does lexical analysis is the scanner

'This is a test with returns in it!
findTokens: (Character or asString), (Character
Sp

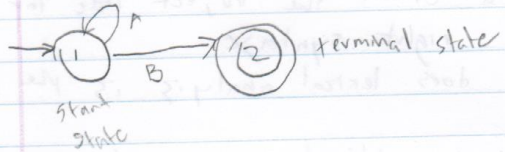
Ordered Collection ('This' 'is' 'a' 'test' 'with' 'returns' 'in' 'it.')

- Recognize when an important token has arrived

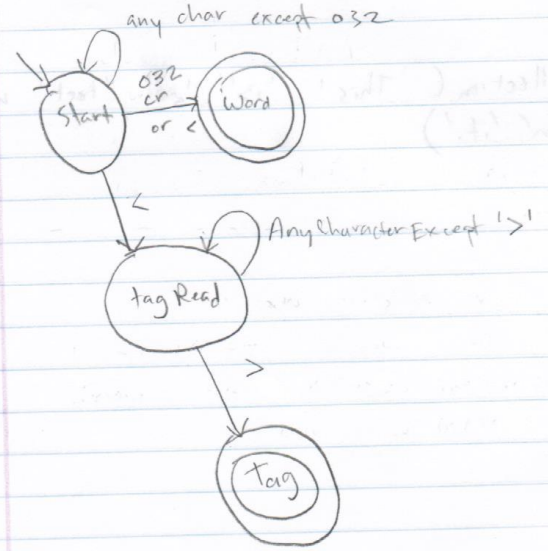
Finite State Automata (FSA)

FSA - has no memory - just a current state

deterministic
FSA - collection of states. Arcs between states



(Incomplete State)



 | bold

read a character; State Trans Table
append the character to our given VALUE;
find the state to move to given our current STATE and the input character;
if it's an end state, push the character back and return the VALUE and the STATE.

Floating Point Number FSA

State	0-9	period
A	A	A
B	B	END

Start	Any Tag But Ø32	Ø32, cr	<	>	Any Character Except>
Start	Start	Word	tagRead	Start	Start
Word	End	End	End	End	End
tagRead	tagRead	tagRead	tagRead	Tag	tagRead
Tag	End	End	End	End	End

Parser - does the syntatic analysis
We will use context-free grammars
- The structure for describing the relationship between tokens is called a grammar.

top level non-terminal → Foo := A | B

Foo := A BAR

BAR := C

AB → Foo

ABA → maps to nothing

AC → Foo

BAR := AFoo

ABA →

Exp := Factor + Expression

Exp := Factor

Factor := Term • Factor

Factor := Term

Term := Number

Term := Identifer (variable)

Document := <html> Header Body </html>

Header := <header> <title> TitleString </title> </header>

TitleString := any alphanumerics plus spaces

Body := <body> BodyPart </body>

BodyPart := any Characteristics

use recursive descent parsing

Expression := Factor + Expression
Expression := Factor

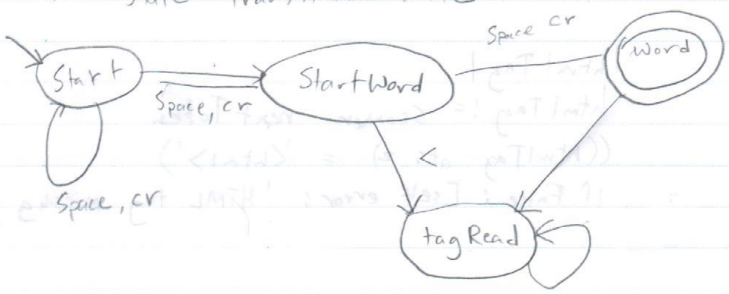
```
expression
    Transcript show: 'Expression'; cr.
    self factor,
    (self peek = '+')
        ifTrue: [Transcript show: '+'; cr.
                self pop.
                self expression. ].
```

Use recursive descent parsing when doing something simple, but use yacc or bison for something big.

```
eqn := EquationParser new.
eqn tokens: ('3 * 4 +5' findTokens: (Character space
    asString)).   " Creates ordered collection => '3' '*' '4' '+' '5'
eqn expression.
```

FSA - Finite State Automata

STT - State Transition Table



A B C D

stream next

B C D

peek

B C D

!!String to Stream!!

tags := Tag Scanner new.
tag stream: (ReadStream on: '<b> a bold statement </b')
5 times Repeat

Parser

instance var 'Scanner'

```
|htmlTag|
htmlTag := Scanner nextToken
((htmlTag at: 2) = '<html>')
if False: [self error: 'HTML tag missing']
```

MVC – Model View Control (79)

<html> ^M
<head>

<html> <head> <title>

May 11, 1999

Designing UI Software

- now modal -
- modal interface - vi

Have a delay to prevent multiple clicks
(Delay for Milliseconds: 500) wait.

Clockbutton make: at: for:

only need to initialize it once. If you use the cheap way, then it will check for the collection each time.

Display Controller

WinController — List Controller
— Paragraph Editor Controller

If you do it right, you can have the "clock" display the analog and digital at the same time.

* The model never sends data to the view unless the view asks for it.

query - what the clockText needs to know about when it needs to update

HTMLParser
HTML Scanner
HTML Viewer — HTML Viewer
HTML Display
Thepage

* Buttons *

* Frame *

May 13, 1999

Strengths and Weaknesses of Model - View - Control

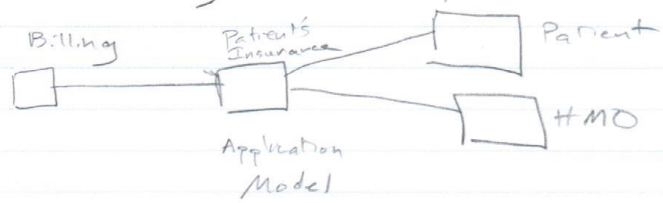
MVC is the dominant metaphor for UI construction today

Strengths

- Clean object - oriented structure. Minimizes information sharing
- Can support multiple views on the same model.

Weaknesses

- Inefficient
- Especially inefficient for multiple views.
- Gets very complicated if you want to have a view drawing from multiple models.



Alternative to MVC - morphic - more flexible but more inefficient, everything exists as a window

Morphic - easy to do animation - all windows and objects act in the same way.

(A4) news.cc.gatech.edu

Simple Client Socket

May 18?, 1999

C++

C - 1920's by Dennis Ritchie at Bell Labs
- Based on BCPL
- Used for writing and maintaing UNIX
- C++ developed in early 1980's by Bjarne Stroustrup
- C programs over about 100,000 lines of code become extremely difficult to read.

- C++ is strongly typed - can't mix types
- Compiled: faster, smaller, less flexible
- Mixed object-oriented and procedural: You can cheat

```
#include <iostream.h>        // import
main () {
    cout << "The volume of the box car is ";
    cout << 11 * 9 * 40;
    cout << endl;
        //
        send chars out to the stream.
}
```

```
#include <iostream.h>
main () {
    int result, height = 11, width = 9;
    int length = 40;
```

```
    char name[10];
```

name →

```
char * name;
  name = malloc (10);


class book {
 public:
   book (char *, char *, int);
   void show_book (void);
 private:
    char title [64];
    char author [64];
    int pages;
};


class catalog_card : public book {
 public:
   catalog-card (char*, char *, int, char *, int);
   void show_card (void);
 private:
   char catalog [64];
   int checked_out;    // 1 if checked out, otherwise 0
};


book :: book (char *newtitle, char *newauthor,
              int newpages) {
    strcpy (title, newtitle);
    strcpy (author, newauthor);
    pages = newpages;
}
```

catalog_card :: catalog_card (char *title, char *author ---)

describe SmallTalk's inheritance.
combination of public and protected.
SmallTalk methods are public.

ISA HASA delegation.

Implementation Inheritance
CLOS, Python

virtual void

Levi D. Smith CS 2390 May 20, 1999

fileStream ⟶ putall // File IO

```
f := FileStream fileNamed: 'aTest'.
f nextPutAll: 'Some Text'; cr.
f nextPutAll: 'Some Text'; cr.
f nextPutAll: 'Some Text'; cr.
```

aTest
Some Text Some Text Some Text

```
|browser|
browser := Scamper new.
```

JIT - Just in Time
Fastest SmallTalk is 2.5 times faster than C
SmallTalk is more portable than Java
most Java is written in C or C++
all SmallTalk is written in SmallTalk
look at the Window shapes
Java - native Windows
SmallTalk - it's own windows
SmallTalk is also slower than Java

Java	C++	SmallTalk (Squeak)
OS (not usually)	Systems Programming	OS (not usually)
RT (claimed)	→ Real Time	RT (claimed)
	→ OS Programming	
	→ Device Drivers	

Real Time Programming - Watch, PDA, cell phones
(time is a critical element)

MYST - hypercard (SLOW)
Director (even slower)

Java	C++	Small Talk (Squeak)
Client DB	Database Server	Client DB
Distributed Apps	Grammar/Spell Checker	Distributed Apps
	DOOM	

May 25, 1999

- What are the users' goals?
- What do they know how to do.
  GOMS analysis
- How often do they try to achieve various goals

- Strive for consistancy
- Provide shortcuts
- Offer Informative feedback
- Design dialogs to yield closure
- Offer Error prevention/handling
  (match pairs, complete sequences, correct commands, provide dialogs)
- Permit easy reversal of actions
- Support internal locus of control
- Reduce short-term memory load
  (seven +/- two)

$$\begin{array}{r} 25 \\ +33 \\ +10 \\ \hline 60 \end{array}$$

- Clock $\xrightarrow[\text{Click}]{\text{MouseOver + ToolTip}}$ Dialog
- Clock + Option Buttons

✗ 12:43 pm [SetTime] [Set Alarm]

✓ 12:43 pm [SetTime] [SetAlarm]

May 27, 1999

Simulations

Descrete Event Simulations

Objects buy you an easy way to make simulations,

When to use computers for simulations,

D | Sim City => a population.
C | Nuclear Weapons testing
C | Enzyme Simulation.
D | Traffic Simulation
C | Material Stresstesting
C | Weather
C | Aerodynamics

D/C MC → Wall Street
D → Factory Floor (Assembly Line)
D → Business (Advertising)
→ Store layout (how many cashier, where to place products, etc.)

Continuous Simulation - You need to simulation at all times
for $t = 0$ to ∞
sim (t)

Discrete Event Simulation - You don't have to simulate every point in time. Only when important events happen; (discrete moments in time).

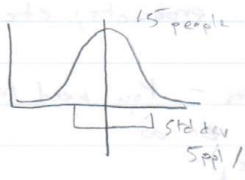
find the area under a curve by using an Integral (continuous) compared to using rectangles (discrete event)

Building Simulations in the Simulation Pkg.st

1. Create a subclass of Simulation to schedule arrivals and set up resources.
2. Create subclasses of Simulation Objects to actually <u>DO</u> things in the simulation.
3. Tell the Simulation instance to proceed for awhile.
4. Gather the statistics you need, perhaps using some of the statistics gathering routines to send things out to disk.

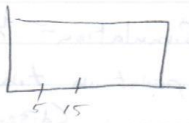
resource → books in a library
(non consumable, non reusable resource)

* Discrete Event Simulations can jump forward in time.



Normal Distribution

std dev ~ 70%



Uniform Distribution

Exponential Distribution

Schedule Arrival Of: a SimulationObject at: timeInteger
// one single object
Schedule Arrival Of: a SimulationObject Class according To:
a Probability Distribution.
// produces lots of objects for you.
- Bomb → atoms scattering

distribution := Normal Mean: 3 derivation: 2.
20 times repeat: [Transcript show: (distribution
next printPrint String); cr]

throughput Histograms.

~~regular resource~~
uncordinated resources (guy asks for a car, one
is available, guy 2 asks for a car, guy 2 takes car,
~~guy~~ car isn't available for guy 1 any more
cordinated resource - how money works in a
simulation - both objects have to be there
at the same time

Important Stuff
Dis - Cont Sim
Resources - Simulation Objects

Tuesday - Research, Final
Thurs June 3, Final Available
Due following Wed.
254 CoC

47
56
95
50

$4\overline{)248}$ 62

62.0
.33
186
1860
20.46

9.8.9 - LABS 10%
62.0 - PROJECTS 33%
84.0 - MIDTERM 25%
- FINAL - 33%

33
25
30
10
98

19.9
20
21
50

2
84
.25
420
168
210

NEED 30% on final for 60
NEED 60% on final for 70
NEED 90% on final for 80

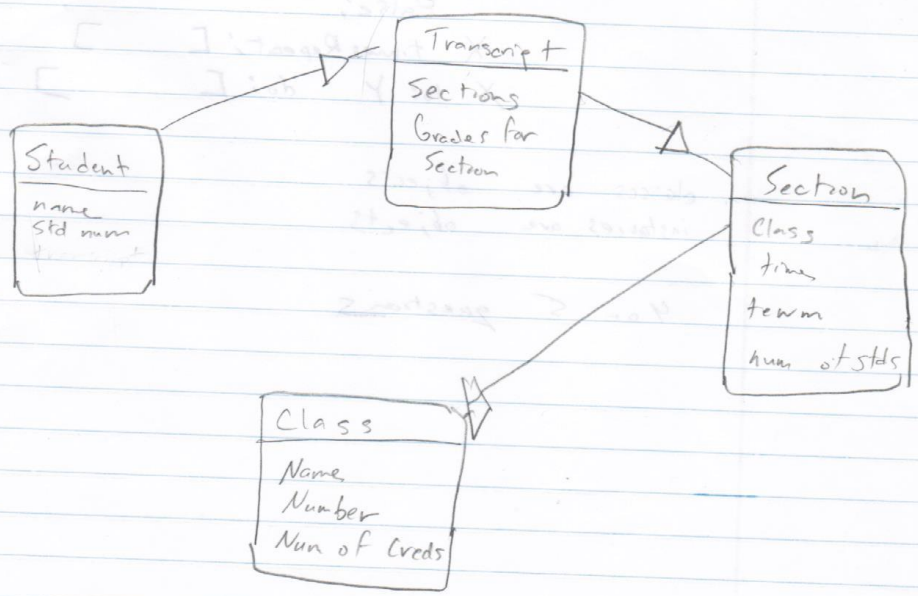
101 - 6:30 PM

1.) lock
door — X ~~connection~~
2.) key — X hotel - name abstraction, reusability
3.) customer — X encoded number - lang specific

A.) Classroom HAS A student

Name — knows name
ID number — knows number

A.) abcdefghijkl
B.) 4
C.) 9



insert key
engine new.

engine
Transcript Show: "Putt"; cr.

△ has-a ⟺ part of
∩ is-a ⟺ parent class / subclass
— ⟺ talks to - communicates with

[condition] whileTrue: [ ]
False:
X timesRepeat: [ ]
X to: Y do: [ ]

classes are objects
instances are objects

4 or 5 questions

Know difference between instance method and class method

Edit this Page Back to the Top

Sp99 Midterm Review

Typically, a midterm in 2390 is 4-5 questions like each of the below. Most of the questions below are from actual midterms in 2390.

I recommend working together to figure out the solutions. I will check up on the discussions, but I won't provide answers where no answers are posted.

Unless you need to edit/repair something here, do not edit this page! Put your answers on the answer page. If you do edit this page, please be careful with the formatting. Thanks! Mark Guzdial

Hotel Doors OOA/D

You have been hired to design a door lock and tracking system for a hotel. These are doors that are opened with a key card that contains an encoded number of some sort. NOTE: The keys have no processor nor network connection. You can't "talk to" a key.

- Each door lock has a connection to a central lock database
- A master key should open any door.
- A customer is given one or more keys that should open the door that she is assigned to
- If a customer loses a key, all the current keys should be made invalid, and only the new keys should be accepted.
- When a customer checks out, all the current keys should be made invalid, and only the new keys should be accepted.

(a). Name at least two classes that you would need in the design for this system, and two classes that you might consider but would reject. Give your reasons for rejecting those two classes.

(b) On the back of this sheet or on a separate sheet that you attach to the exam (remember, each problem will be graded separately!), draw the CRC cards for this scenario:

> **A customer gets a card key, takes it to the door, puts it in the door lock which checks if it's valid, then the door opens because it is valid.**

Put your answers/comments/questions at Sp99 Midterm Review: Hotel Doors OOA/D

Analyze Student/Classroom Code

Imagine that I have a Student object in a Classroom object, something like this.

```
| s c |

c := Classroom new name: 'CS2390'.
s := Student new name: 'Fred'; section: 'a'; number: '123456789'.

c add: s.
s := Student new name: 'Wilma'; section: 'b'; number: '3456789012'.

c add: s.
s := Student new name: 'Barney'; section: 'a'; number: '012345678'.

"...More code here..."
```

a. (7 points) What would you say is the relationship between Student objects and the Classroom object is? Why?

b. (10 points) What attributes and services do you think that the Student and Classroom objects have, based on what you see in the above code?

Put your answers/comments/questions at Sp99 Midterm Review: Analyze Student/Classroom Code

---

Short Essays

sims - specific
OO - more reusable

a. Explain how object-oriented programming is related to simulations. - the both try to model the real world - many objects of the same type, (by using objects)

b. What is a CRC card? What are they good for?

Class / Responsibility / Collaborator
good for design - lets you see how classes interact - brainstorming

c. Explain in what sense inheritance is a form of delegation.

Expanding the work out between classes,

d. Why is it useful to think about objects as "biological cells"? (Hint: Think about scalability and about modularity.) cells ~~to~~ have properties and know their responsibility
scalability - can make a lot of identical objects - also the cells work together as an object must do

Put your answers/comments/questions at Sp99 Midterm Review: Short Essays

---

Reading Code

Transcript show: 'Hello World'

4 times Repeat: [ ]

[ ] whileTrue: [ ]

```
| anArray aString |
anArray := #('abc' 'def' 'ghi','jkl').
aString := ''.

anArray do: [:each |

        aString := aString, each.].

Transcript show: aString.
```

A) abcdefghi, jkl

B) 5 times

a. What appears on the `Transcript` ?

b. How many times did the `do:` loop get executed?

[conditional] whileTrue: [do this]

```
| i test |

i := 1.

test := (i < 10).[test] whileTrue: [Transcript show: 'hello'.   i := i + 1.].
```

this is tested

only gets evaluated once

c. How many times does `hello` get printed in this example? c) infinite

Put your answers/comments/questions at Sp99 Midterm Review: Reading Code

---

Definitions

Define each and give an example.

a. What is Inheritance? -

b. What is Delegation? - delegating responsibility - don't have one class do all of the work

c. What is Polymorphism? - write the code to deal with A, but at runtime the class type is checked and appropriate method for that class is called.

A → B, C

| A | A | B | C |

Put your answers/comments/questions at Sp99 Midterm Review: Definitions

---

Draw a Class Diagram of a Student Registration System

You have just completed an analysis of a student registration system:

- Students have names, student numbers, and a transcript
- Transcripts list the sections that a student has taken and the grade for each.
- A Section is an offering of a Class that has a time and term associated with it, and a number of

Sp99 Midterm Review

students who took the class that quarter.
- A Class has a name, a number (e.g. "CS2390"), and a number of credits associated with it.

Draw a class diagram for these classes. You can upload it to the 2390 Warehouse and then link it in.

Put your answers/comments/questions at Sp99 Midterm Review: Draw a Class Diagram of a Student Registration System

---

Simulate a Motorcycle

Imagine an object-oriented simulation of a motorcycle.

- When the key is inserted into the ignition, the `key` object tells the `engine` object to `start`.
- The engine should then start to run. When the `engine` object is running, it prints **Putt!** to the `Transcript` once a second.
- Each time the `engine` objects "putts," it sends a message to its `wheel` object to `turn`.
- When a `wheel` object turns, it prints **Turn!** to the Transcript.

What instance variables are you going to need for each of the key, engine, and wheel classes?

(b) Write the methods insert for key; start and run for engine; and turn for the wheel.

(c) Write whatever accessor methods you need to make the above example work.

(d) Write the workspace code that you'd use to create the above scenario

Put your answers/comments/questions at Sp99 Midterm Review: Simulate a Motorcycle

---

Explain the Metaphor

a. What does it mean "objects are like cells"?

b. What does it mean that "objects are like small computers inside the computer"?

Put your answers/comments/questions at Sp99 Midterm Review: Explain the Metaphor

---

Drawing in Smalltalk

1. Using Pen, write the code to draw an equilateral triangle.

2. Using the Box class, write the code to draw a pyramid of three boxes.

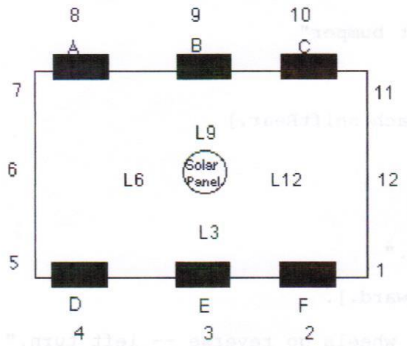
3.Explain what this code will do:

```
| joe |
joe := Box new.
30 timesRepeat: [joe turn: 12.]
```

Put your answers/comments/questions at Sp99 Midterm Review: Drawing in Smalltalk

Analyze Robot Car

A robot is being designed to explore the surface of Mars autonomously.



Features of this robot include:

- Six wheels A-F (three per side) which can go forward or backward independently. There is a Controller that can also activate all wheels at once.
- Two sets of sensors. Sensors can be on or off.
- An array of twelve distance sensors 1-12 around the perimeter. Distance sensors report distance to nearest obstacle in direction of sensor.
- An array of four light sensors pointed upward and outward: L3, L6, L9, and L12. Light sensors report brightness.
- A solar panel on an axis and motor capable of turning 360 degrees.

You are being asked to write some of the operating software for this robot.

Create a class diagram for the robot.

(You don't have to add more than what's in the above example.) Note: Partial credit will be given, so err on the side of too much, not too little. In your analysis, be sure to address:

- What classes do you need? What service do each need to provide? What attributes do each need to know?
- Where are there generalization-specialization relationships? Where are there part-whole

relationships?

Put your answers/comments/questions at Sp99 Midterm Review: Analyzing the Robot Car (You can upload diagrams to 2390 Warehouse

Code for Robot Car

Code for Robot Car: Here is part of the vehicle avoidance method for the robot:

```
moveBackAfterBumpingFront: sensor

        sensor = 11

        ifTrue:

        [ "Assume that welve bumped our front left bumper"

        "Shift all wheels into reverse gear"

        #('a' 'b' 'c' 'd' 'e' 'f') do: [:each | each shiftRear.]

        "Activate all wheels for one second."

        wheelsController activateFor: 1.

        "Shift left side wheels into forward gear."

        #('a' 'b' 'c') do: [:each | each shiftForward.].

        "Makes left wheels go forward while right wheels go reverse -- left turn."

        wheelsController activateFor: 1.

        ].
```

Write the analagous code for bumping the front right bumper.

```
        sensor = 1 ifTrue:

        [

Your code goes here

        ]
```

Put your answers/comments/questions at Sp99 Midterm Review: Coding the Robot Car

Edit this Page
Search for References to this Page.

Search for text in all pages: Start Search

Display this Page and all its References

CS 2390 - Assignment 1 Grading

NAME: Levi Smith

Diagram	20	/ 20
CRC	20	/ 20
Classes involved included	10	/ 10
OOA/OOD - Overall Quality	15	/ 15
Scenarios	0	/ 5
Reuse	10	/ 10
Free Points	20	/ 20
OVERALL TOTAL	95	/ 100

Levi D. Smith
April 14, 1999
CS2390 C

OOA

OOA Discussion

This is the design for a program to manage various kinds of media. This design monitors newsgroups, webpages, and cowebpages. When the user logs on, this program will check all of the webpages and cowebpages in the user's list, and will inform the user if any of the pages have been updated. The program determines if a page has been updated by comparing the size to the size specified in the storage file. This process is simpler for the coweb pages since all coweb pages are time stamped. Also, this program will search all links on one of user's pages for updates. Newsgroups are also checked for new posts. When the user logs on, the program will search through all posts and return the ones which have a date later than the date when the user last logged started this program.

Webpages have been sorted into four categories to make browsing them easier for the user.
1 ClassWebPages are for pages that deal with pages that contain information about the user's classes.
2 OrganizationWebPages are for the clubs and organization that the user may belong. Finally, personal pages are for the user's friends and family and attributes about them. 3
What's the fourth category?
I have many rejected classes. I found that most of them were really attributes to other classes rather than classes themselves. I also had a SystemMonitor class. This was good in that it allowed several instances of the monitor program for many users. On the other hand, it created one central controlling "god" class, so I removed it from list of classes.

No Candidate Classes

Rejected Classes:

- HTMLSource
- HTMLTags
- NewPost
- NonUpdatedPage
- SubscribedNewsgroups
- System Monitor
- URLLimit
- UnsubscribedNewsgroups
- UpdatedPage
- WebpageAuthor
- WebpageContent
- WebpageTitle

OOA CRC Cards

ClassWebPage	
Superclass: WebPage	
Responsibility	**Collaborator**
hold important data for this class	
inform students of activities	

CoWeb Monitor	
Responsibility	**Collaborator**
Keep a list of all CoWeb sites	CoWebPage
make a list of updated pages	CoWebPage

CoWebPage	
Superclass: WebPage	
Responsibility	**Collaborator**
allow many to add to a website	

NewsGroup	
Responsibility	**Collaborator**
Report new posts	
keep posts in an ordered fashion	

NewsgroupPost	
Responsibility	**Collaborator**
Report date of posts	NewsGroup
inform people of current news	

TOO MANY CLASSES!

I understand what you're doing, and it makes sense, but this is a massive undertaking (at least, programming-wise). Especially for webpages - should just need one generic webpage class. Only need to know when/if it was updated. Other info is superfluous. Not that this is a bad thing - just saying to make things easier for you if ya can.

Organization/Club WebPage	
Superclass: WebPage	
Responsibility	**Collaborator**
alert members of events	

PersonalWebPage	
Superclass: WebPage	
Responsibility	**Collaborator**
let you know what your friends and family are doing	

StorageFile	
Responsibility	**Collaborator**
keep a record of all media visited	

Newsgroup Monitor	
Responsibility	**Collaborator**
keep a list of all user's newsgroups	NewsGroup
make a record of updated newsgroups	NewsGroup

WebLinks	
Responsibility	**Collaborator**
allow user to easily move through pages	

WebPage	
Subclasses: ClassWebPage, CoWebPage, Organization/Club WebPage, PersonalWebPage	
Responsibility	**Collaborator**
store information and data	

WebSite Monitor	
Responsibility	**Collaborator**
keep a record of all websites	WebPage
make a record of updated sites	WebPage
find updated sites	StorageFile

OOD

ClassWebPage
Superclass: WebPage
Attributes • ClassName • ClassNumber • ClassSection • Instructor's Name • Quarter/Semester • School/College Name
Services • display information on this class

Do we really need this? I don't know - possibly

CoWeb Monitor
Has Parts: CoWebPage
Attributes • Last Time Accessed • updated CoWeb Pages
Services • find newCoWebSites

CoWebPage
Superclass: WebPage Part of: CoWeb Monitor
Attributes • dateEdited • editedBy • hasBeenEdited
Services • report edit date

NewsGroup
Part of: Newsgroup Monitor Has Parts: NewsgroupPost
Attributes • Address • Name
Services • determine if updated • find new posts

NewsgroupPost
Part of: NewsGroup
Attributes • AddressedToYou • Carbon Copies • Replies/FollowUps • date • posted by • read/unread

Services

- return if new posts exist

Organization/Club WebPage
Superclass: WebPage
Attributes • Meeting Times • OrganizationName • club services
Services • report club activities

PersonalWebPage
Superclass: WebPage
Attributes • FamilyMember/Friend • Person's Name
Services • report creators relation

StorageFile
Attributes • StorageData
Services • keep data of current session

Newsgroup Monitor
Has Parts: NewsGroup
Attributes • Last Time Accessed • NewsGroupReader • Updated Newsgroups
Services • report updated newsgroups

WebLinks
Part of: WebPage
Attributes • URL • foreign link/ link on this domain • http/telnet/ftp/etc. • name
Services • handle links pages

WebPage
Subclasses: ClassWebPage, CoWebPage, Organization/Club WebPage, PersonalWebPage Part of: WebSite Monitor Has Parts: WebLinks
Attributes • Author/Creator • Content • HTML/XML/VRML • Title

- usesFrames
- usesJavascript

Services

- crawl through weblinks
- determine if updated

WebSite Monitor
Has Parts: WebPage
Attributes • Last Accessed • URL Limit • Updated Websites • WebBrowser
Services • find updated Websites

OOD Discussion

Monitor Implementation: each montior is responsible for keeping track of which websites/newsgroups have been updated. A better design might include only one monitor, instead of having one for each medium. Each monitor then returns it's list of updated sources to the interface.

WebPage Implementation: webpages will be separated according to their type (personal, class, coweb, or organization). This will allow easier browsing for the user. They will be compared to the storage file to determine if it has been updated.

Newsgroup Implementation: each of the posts will be searched by date. if the date on the post is later than the time when the user last logged onto the program.

Interface: the data from the WebSite Monitor, CoWeb Monitor, and Newsgroup Monitor will be grouped together and displayed to the user one screen, or the user may be given the choice to view each separate list of updated media. When the user clicks on one of the links to the medium, then a web browser or mail client will open taking the user directly to the document they selected.

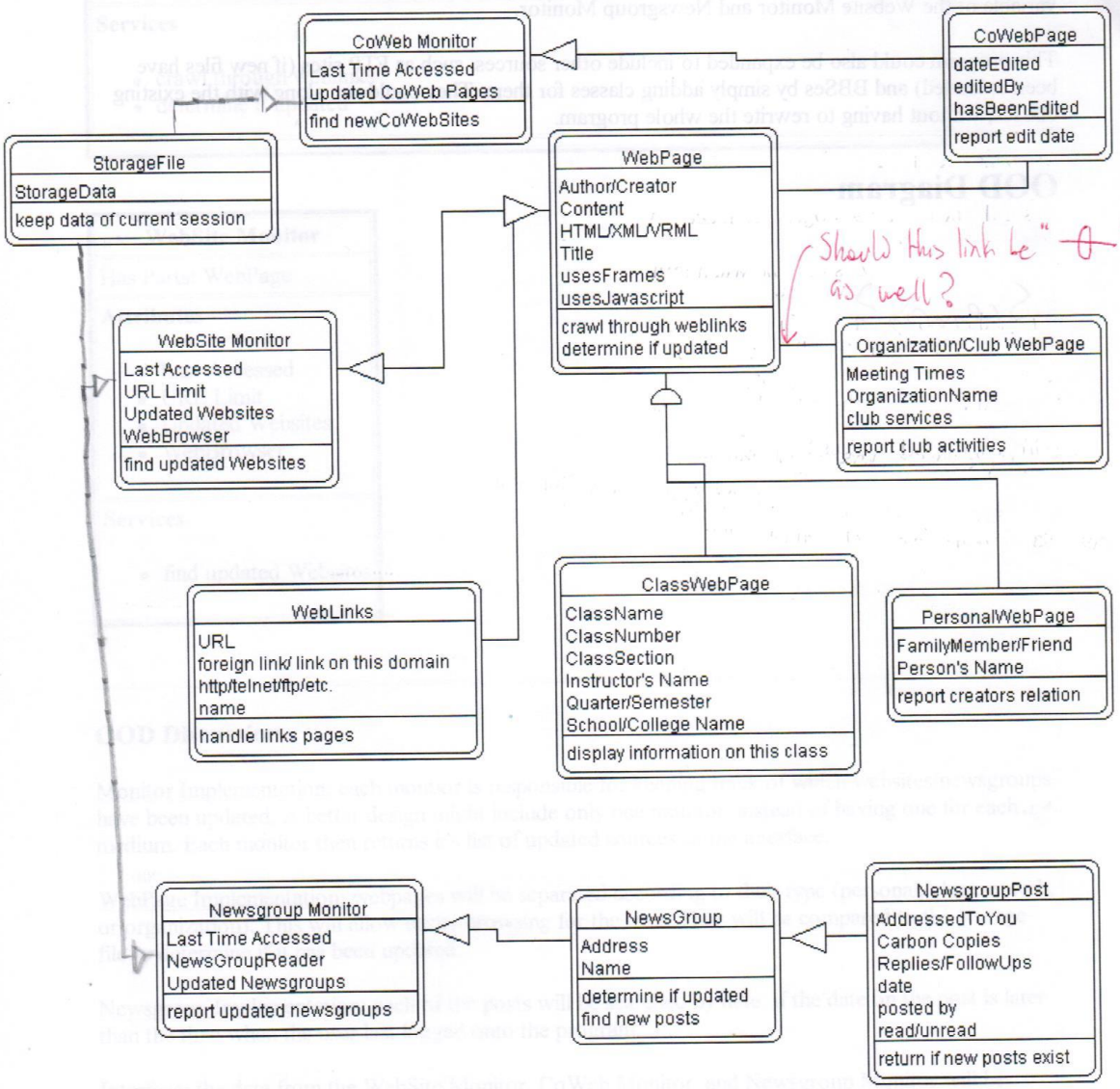
Reusability

Instead of having just one list of links and newsgroups to search for updates, many users could use this program, and each could have his own list of website/newsgroups by simply making another instance variable of the Website Monitor and Newsgroup Monitor.

This program could also be expanded to include other sources, such as FTP sites (if new files have been uploaded) and BBSes by simply adding classes for them. They could run along with the existing classes, without having to rewrite the whole program.

OOD Diagram

Scenario?


CoWeb Monitor
Last Time Accessed
updated CoWeb Pages
find newCoWebSites
CoWebPage
dateEdited
editedBy
hasBeenEdited
report edit date
StorageFile
StorageData
keep data of current session
WebPage
Author/Creator
Content
HTML/XML/VRML
Title
usesFrames
usesJavascript
crawl through weblinks
determine if updated
Should this link be "θ" as well?
WebSite Monitor
Last Accessed
URL Limit
Updated Websites
WebBrowser
find updated Websites
Organization/Club WebPage
Meeting Times
OrganizationName
club services
report club activities
WebLinks
URL
foreign link/ link on this domain
http/telnet/ftp/etc.
name
handle links pages
ClassWebPage
ClassName
ClassNumber
ClassSection
Instructor's Name
Quarter/Semester
School/College Name
display information on this class
PersonalWebPage
FamilyMember/Friend
Person's Name
report creators relation
Newsgroup Monitor
Last Time Accessed
NewsGroupReader
Updated Newsgroups
report updated newsgroups
NewsGroup
Address
Name
determine if updated
find new posts
NewsgroupPost
AddressedToYou
Carbon Copies
Replies/FollowUps
date
posted by
read/unread
return if new posts exist

Assignment 3

Discuss at http://pbl.cc.gatech.edu/cs2390/1333.html. Form your team or discuss with your team at http://pbl.cc.gatech.edu/cs2390/1332.html. Due date on Class Schedule

Design and Implement an HTML Page Viewer

YOU MUST WORK IN TEAMS OF TWO TO FOUR ON THIS PROJECT! The cost of not working on a team is an immediate 15 point deduction on your grade.

When you turn Assignment 1 in, by the end of class, I want a sheet of paper from each group listing the group members, their email addresses, and phone numbers. You can use the team page at http://pbl.cc.gatech.edu/cs2390/1332.html to advertise for your team.

I want you to write Smalltalk code to:

- Read an HTML page from an http:// or file:// URL
- Display the page for a subset of HTML in a 640x480 area on the screen
- Provide scroll buttons at the top and bottom to scroll down (half-page) or back up (if already scrolled down.)

The subset of HTML that you will support will be:

- Breaks:
- Paragraph marks: <p>
- Bold, italics, and underline: <b>, <i>,
- Center:
- Horizontal rules: <hr>
- Headings following these rules:
  - <h1> are all uppercase and underline and boldface
  - <h2> are all uppercase and boldface
  - <h3> are all uppercase
  - All other levels are just displayed as a separate paragraph
- Anchors, but not with clickable links. Just display the anchor text in italics WITH THE COMPLETE REFERENCED URL (not just relative links) IN PARENTHESES AFTER THE ANCHOR TEXT. some anchor text

For extra credit (5 points), you can support images (GIFs, JPEGs, and BMPs), just displayed in-line, no sizing or formatting: <image src="http://somewhere/something.gif">. If you don't support images, just display the word IMAGE in its own paragraph at the image marker.

In doing this project, you may not use any windows, menus, buttons, or text areas built-in to Squeak. You must display graphics directly to the screen (suggestion: clear out a 640x480 area with white, first), and you read the Sensor to determine if the mouse has been clicked and where.

I want to use your code like this:

```
HTMLViewer on: 'http://www.cc.gatech.edu/index.html'
```

User Interface

You must provide a way to scroll up and down long pages. It is up to you to come up with a usable scroll system. Buttons at the top and bottom? Scroll bar with "thumb"? You decide, but you have to post the graphics yourself, do your own event loop, and handle scrolling. Scrolling should scroll about 1/2 page at a time.

Implementation Hints

- The tricky part here is probably not going to be the formatting, but the word wrapping. How do you know when to wrap a line and start a new line?
- Most of what you need to display formatted text is in the Text class. Look at the various supports for attributes and the classes TextAttribute, TextFontReference, TextFontChange.
- Text's know how to display strings with attributes, but the actual display (e.g., where on the screen, how much space on the screen) is handled in DisplayText.
- To get the size of a displayed text chunk, explore the below code:

```
www _ DisplayText text: ('www' asText) textStyle: TextStyle default copy cente
lll _ DisplayText text: ('lll' asText) textStyle: TextStyle default copy cente
www display.
lll display.
Transcript show: (www form boundingBox printString) ; cr.
Transcript show: (lll form boundingBox printString); cr
```

Scenarios

- What happens if someone goes to a page with frames or tables? Suggestion: Ignore them gracefully, no crashes.
- How do you display a large GIF on one page, and three pages of text on the next, and then what do you do when someone scrolls forward one page?

Grading

You will need to turn in your Design Documentation which includes your code, and the printout from above -- ONE PER GROUP. You can turn in to any section that you have a member from. **You must also turn in the statistics with average time to process in your design documentation**. EACH GROUP MEMBER will also need to mail your code to cs2390@prism.gatech.edu with the subject line "a3, 123456789,d" where 123456789 is your student number and "d" is your section (a, b, c, or d).

CRITERIA:

OOA/D (34 total)

- Good, detailed, REUSABLE Design - 18 (9 points for good HTML/parsing objects, 9 points for UI objects)
- CRC Cards - 8 points
- Diagram - 8 points

Scenarios 6

(3 points each)

Reuse (believeable) 9

OOP (51)

- HTML parsing
  - 5 Handles breaks and paragraphs
  - 10 Handles bold, italics, and underline
  - 3 Handles centering
  - 3 Handles three levels of headers
  - 5 Handles horizontal rules
  - 10 Handles anchors and complete URL references
- UI
  - 5 Displays good, usable, readable scroll buttons
  - 10 Scrolls 1/4 page each way
- 5 points extra credit for correct display of IMAGE tags

Back to the top | Over to CoWeb

CS 2390 - Assignment 3 Grading

NAMES: Charles Alderman gt46520 sec B Tim Sotack gte283k sec B

Dave Brown B Levi Smith gte187k

Diagram	8	/ 8
CRC	8	/ 8
OOA/OOD - Overall Quality	16	/ 18
Scenarios	6	/ 6
Reuse	9	/ 9
HTML - Handles breaks & paragraphs	0	/ 5
HTML - Handles bold, italics, underline	0	/ 10
HTML - Handles centering	0	/ 3
HTML - Handles headers (3 levels)	0	/ 3
HTML - Handles horizontal rules	0	/ 5
HTML - Handles anchors, references	0	/ 10
UI - Displays usable scroll buttons	0	/ 5
UI - Actually scrolls	0	/ 10
Extra Credit - Handles Image Tags	0	/ 5
OVERALL TOTAL	47	/ 100

Assignment3

Charles Alderman, gt4652d
Tim Sotack, gte283k
Dave Brown, gt7241a
Levi Smith, gte187k

May 12, 1999

OOA

No Candidate Classes

No Rejected Classes

OOA CRC Cards

ViewerButton	
Responsibility	**Collaborator**
draws itself	
Accept user input to scroll up or down	

HTMLParser	
Responsibility	**Collaborator**
gets tokens	HTMLScanner
Ability to change state according to tag	
Tell HTMLviewer to display tokens of the specified tag	HTMLviewer
Return an error for unmatching tags	

HTMLScanner	
Responsibility	**Collaborator**
Break stream into tokens	
Take in a filestream	HTMLviewer

HTMLviewer	
Responsibility	**Collaborator**
create display	
Display text according to a limited HTML	HTMLParser
Scroll the text on the display according to user input	ViewerButton
Take in a URL and grab the specified file	ThePage

ThePage	
Responsibility	**Collaborator**
Loads page from URL	
Returns contents of page	

What needs to be done?
We are designing a viewer to display a limited set of HTML.

Brainstormed Objects:
HTMLviewer
HTMLscanner
HTMLparser
ThePage
ViewerButton

Class HTMLviewer - Grabs a file from a URL, displays the HTML, scrolls the HTML up and down.
Class HTMLscanner - Takes in a file stream, breaks the stream into tokens.
Class HTMLparser - Takes in tokens, changes state according to token, tells HTMLviewer what to display with specified HTML tag, returns an error for unmatched tags.
Class ThePage - Takes in a URL, loads the contents of the file, returns the file as stream.
Class ViewerButton - Draws itself, accepts user input to scroll the HTML

OOD

ViewerButton
Part of: HTMLviewer
Attributes • Name • position
Services • init • pressed

HTMLParser
Sends Messages To: HTMLScanner, HTMLviewer
Attributes • scanner • theHTMLviewer
Services • body • bodyPart • bold • document • h1 • h2 • h3 • header • image • italic • underline

HTMLScanner

Receives Messages From: HTMLParser

Attributes

- nextCharacter
- peekedToken
- stream
- value

Services

- next
- nextToken
- peek
- peekToken
- stream:aStream
- tag
- tagRead
- word

HTMLviewer

Has Parts: ThePage, ViewerButton

Receives Messages From: HTMLParser

Attributes

- centeredLine
- centeredWidth
- fileURL
- linesize
- spacewidth
- thePage
- winheight
- winwidth
- wordOffset
- wordposition
- wordstart
- xoffset
- yoffset

lots of functionality here
should just be doing viewing
can break this up into
(3?) classes

Services

- control
- displayBoldWord:word
- displayBreak
- displayCenter:word
- displayHorizRule
- displayImage:
- displayTitle:pageTitle

- displayTitle:pageTitle
- displayWord:word
- displayWordH1:word
- displayWordH2:word
- displayWordH3:word
- displayWordUnderline:word
- fileURL:
- initialize:
- makeForm
- on:URL

ThePage
Part of: HTMLviewer
Attributes • pageContents • pageURL
Services • contentsOf • loadPage • pageURL • sizeof

OOD Discussion

HTMLviewer - Takes in a URL, tells ThePage to get the file, takes the file, sends it to the HTMLscanner. Creates a display, takes in what to display from HTMLparser. Instantiates ViewerButtons to determine when to scroll the HTML.

HTMLparser - Takes in the tokens sent by HTMLscanner, changes state according to current tag, sends HTMLviewer text to be displayed with each tag's attributes.

HTMLscanner - Takes in a stream that is the HTML file from HTMLviewer, breaks the stream into tokens, sends the tokens to HTMLparser.

ThePage - Takes in a URL, grabs it from wherever, returns the contents of the file to HTMLviewer.

ViewerButton - Displays itself on the HTMLviewer display. Accepts user input to move the HTML around in the viewer.

Scenarios

What happens if someone goes to a page with frames or tables? Suggestion: Ignore them gracefully, no crashes.

HTMLscanner will break up the file stream into tokens. The tokens will be sent to HTMLparser. If

MLparser recognizes that the token is a frame or table tag, then it will place itself into a frame or table ate and ignore tokens until it receives the end table or frame tag. The rest of the page will be displayed s if the table did not exist.

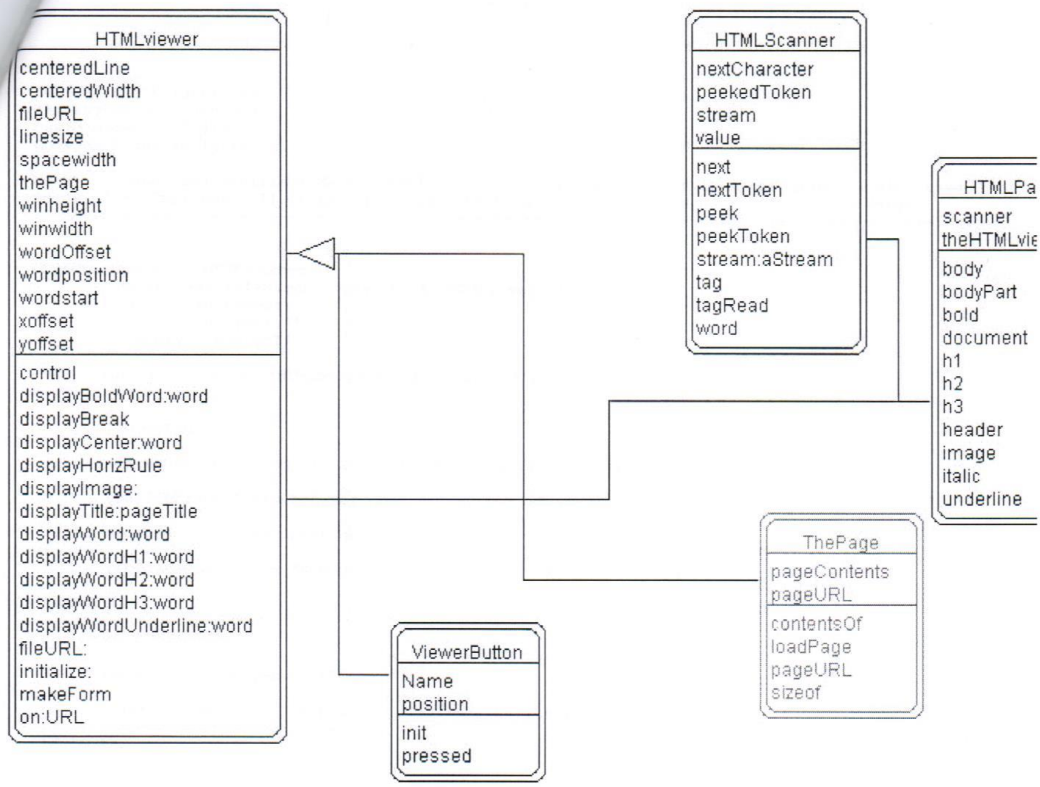
How do you display a large GIF on one page, and three pages of text on the next, and then what do you do when someone scrolls forward one page?

HTML scanner will break the file stream into tokens. The tokens will be sent to HTML parser. If HTMLparser recognizes an image tag then it sends the text "IMAGE" to the viewer since images have not yet been implemented. The rest of the page's text, tags, whatever will then be displayed. When the user hits the scroll down button, then HTMLviewer receives a mesage from the down ViewerButton to scroll down, and it redraws the page halfway down.

Reuse

We want to design an email viewer like Eudora. It will have classes to send, receive, and display emails. Since some emails are encoded in HTML the HTML encoded emails will be able to be displayed. HTMLviewer will be sent the email message that is encoded in HTML, HTMLscanner will then break up the email message into tokens, and HTMLparser will send the parsed HTML to the HTMLviewer that is displaying said email.
Another functionality in email programs is the ablity to attach a URL to an email. Here we can use ThePage to grab the file and put it into the email.

OOD Diagram


HTMLviewer
centeredLine
centeredWidth
fileURL
linesize
spacewidth
thePage
winheight
winwidth
wordOffset
wordposition
wordstart
xoffset
yoffset
control
displayBoldWord:word
displayBreak
displayCenter:word
displayHorizRule
displayImage:
displayTitle:pageTitle
displayWord:word
displayWordH1:word
displayWordH2:word
displayWordH3:word
displayWordUnderline:word
fileURL:
initialize:
makeForm
on:URL
HTMLScanner
nextCharacter
peekedToken
stream
value
next
nextToken
peek
peekToken
stream:aStream
tag
tagRead
word
HTMLPa
scanner
theHTMLvie
body
bodyPart
bold
document
h1
h2
h3
header
image
italic
underline
ThePage
pageContents
pageURL
contentsOf
loadPage
pageURL
sizeof
ViewerButton
Name
position
init
pressed

Levi D. Smith (gte187k)
Timothy Sotack (gte283k)
David Brown (gt7241a)
Charles Alderman (gt4652d)

Note: I used (and modified) Mark Guzdial's Parser and Scanner code in my program. I also used parts of my Assignment II in this program (the location and loading of http and file URLs).

```
Object subclass: #HTMLParser
        instanceVariableNames: 'scanner theHTMLviewer '
        classVariableNames: ''
        poolDictionaries: ''
        category: 'AssignIII'!

!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/12/1999 23:42'!
body

        | bodytag  |

        ((scanner peekToken at: 2) beginsWith: '<frame') ifTrue:

        [self error: 'This browser does not support frames'].

        bodytag := scanner nextToken.

        ((bodytag at: 2) = '<body>') ifFalse: [self error: 'Body tag

missing'].

        self bodyPart.

        bodytag := scanner nextToken.

        ((bodytag at: 2) = '</body>') ifFalse: [self error: 'Body end tag

missing'].

! !

!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/13/1999 00:02'!
bodyPart

        [(scanner peekToken at: 2) = '</body>'] whileFalse:

                [(scanner peekToken at: 2) isNil ifTrue: [self error: 'No

end body'].


        ((scanner peekToken at: 2) = '<b>')

        ifTrue: [

                self bold.

        ] ifFalse: [

                ((scanner peekToken at: 2) = '<u>')

                ifTrue: [

                        self underline.

                ] ifFalse: [

                        ((scanner peekToken at: 2) = '<i>')

                        ifTrue: [

                                self italic.

                        ] ifFalse: [
```

```
((scanner peekToken at: 2) = '<br>')
ifTrue: [
        theHTMLviewer displayBreak.
        scanner nextToken at: 2.
] ifFalse: [
        ((scanner peekToken at: 2) =
'<hr>')
        ifTrue: [
                theHTMLviewer
displayHorizRule.
                scanner nextToken at: 2.
        ] ifFalse: [
                ((scanner peekToken at: 2)
= '<h3>')
                ifTrue: [
                        self h3.
                ] ifFalse: [
                        ((scanner
peekToken at: 2) = '<h2>')
                        ifTrue: [
                                self h2.
                        ] ifFalse: [
                                ((scanner
peekToken at: 2) = '<h1>')
                                ifTrue: [

self h1.
                                ] ifFalse:
[

                                ((scanner
peekToken at: 2) = '<p>')
                                ifTrue: [

self paragraph.
                                ] ifFalse:
[
```

```
((scanner
peekTokenat: 2) = '<table>')
ifTrue: [
self table.
] ifFalse:
[
((scanner
peekTokenat: 2) beginsWith:'<img')
ifTrue: [
self image.
] ifFalse:
[
((scanner peekTokenat: 2) = '<center>')
ifTrue: [
self centered.
]
ifFalse: [
((scanner peekTokenat: 2) beginsWith: '<a href')
ifTrue: [
self link.
] ifFalse: [
"default case"
theHTMLviewer displayWord:
(scanner nextTokenat: 2).
].
].
].
].
].
].
].
].
].
].
].
].
].
].
].
].].].! !
```

```
!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/9/1999 00:51'!
bold

(scanner nextToken at: 2). "Strip the first <b> tag"

	[(scanner peekToken at: 2) = '</b>'] whileFalse:

		[(scanner peekToken at: 2) isNil ifTrue: [self error: 'No

end bold'].


		theHTMLviewer displayWordBold: (scanner nextToken at: 2)].

(scanner nextToken at: 2). "Strip the closing </b> tag"! !

!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/9/1999 21:39'!
centered

scanner nextToken.


	[(scanner peekToken at: 2) = '</center>'] whileFalse:

		[(scanner peekToken at: 2) isNil ifTrue: [self error: 'No

end center'].


		theHTMLviewer displayCenter: (scanner nextToken at: 2)].

theHTMLviewer displayCenter: (scanner nextToken at: 2). "tell

displayCenter to stop and print all of its contents"

(scanner nextToken at: 2). "Strip the end tag"! !

!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/9/1999 20:49'!
h1

"uppercase"

(scanner nextToken at: 2). "Strip the start tag"


	[(scanner peekToken at: 2) = '</h1>'] whileFalse:

		[(scanner peekToken at: 2) isNil ifTrue: [self error: 'No

header'].


		theHTMLviewer displayWordH1: (scanner nextToken at: 2)].

(scanner nextToken at: 2). "Strip the end tag"! !

!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/9/1999 20:49'!
h2

"uppercase"

(scanner nextToken at: 2). "Strip the start tag"


	[(scanner peekToken at: 2) = '</h2>'] whileFalse:

		[(scanner peekToken at: 2) isNil ifTrue: [self error: 'No
```

```
header'].

                theHTMLviewer displayWordH2: (scanner nextToken at: 2)].

(scanner nextToken at: 2). "Strip the end tag"! !

!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/9/1999 01:20'!
h3

"uppercase"

(scanner nextToken at: 2). "Strip the start tag"


        [(scanner peekToken at: 2) = '</h3>'] whileFalse:

                [(scanner peekToken at: 2) isNil ifTrue: [self error: 'No

header'].


                theHTMLviewer displayWordH3: (scanner nextToken at: 2)].

(scanner nextToken at: 2). "Strip the end tag"! !

!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/13/1999 00:02'!
image


(scanner nextToken at: 2). "Get rid of the tag "


                theHTMLviewer displayWord: '[IMAGE]'.

! !

!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/9/1999 01:06'!
italic

(scanner nextToken at: 2). "Strip the first <i> tag"

        [(scanner peekToken at: 2) = '</i>'] whileFalse:

                [(scanner peekToken at: 2) isNil ifTrue: [self error: 'No

end italic'].


                theHTMLviewer displayWordItalic: (scanner nextToken at:

2)].

(scanner nextToken at: 2). "Strip the closing </i> tag"! !

!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/13/1999 00:00'!
link

| theLink |


theLink := (scanner nextToken at: 2).

theLink := theLink copyFrom: 10 to: ((theLink size) - 2).

theLink := ((theHTMLviewer getPage) pageURL), '/', theLink.

"Transcript show: theLink."
```

```
	[(scanner peekToken at: 2) = '</a>'] whileFalse:
		[(scanner peekToken at: 2) isNil ifTrue: [self error: 'No
end anchor'].

		theHTMLviewer displayWordItalic: (scanner nextToken at:
2)].
(scanner nextToken at: 2). "Strip the closing </a> tag"
	theLink := '(', theLink, ')'.
		theHTMLviewer displayWord: theLink.! !
!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/12/1999 22:55'!
paragraph

	theHTMLviewer displayBreak.
(scanner nextToken at: 2). "Strip the first <p> tag"
	[(scanner peekToken at: 2) = '</p>'] whileFalse:
		[(scanner peekToken at: 2) isNil ifTrue: [self error: 'No
end paragraph'].

		theHTMLviewer displayWord: (scanner nextToken at: 2)].
(scanner nextToken at: 2). "Strip the closing </p> tag"
	theHTMLviewer displayBreak.! !
!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/12/1999 23:45'!
table
(scanner nextToken at: 2). "Strip the first <table> tag"
	"since we don't have to display tables, I'm going to delete
everything inside the table"
	[(scanner peekToken at: 2) = '</table>'] whileFalse:
		[(scanner peekToken at: 2) isNil ifTrue: [self error: 'No
end table'].

	(scanner nextToken at: 2)].
(scanner nextToken at: 2). "Strip the closing </table> tag"! !
!HTMLParser methodsFor: 'HTMLbody' stamp: 'lds 5/9/1999 01:07'!
underline
(scanner nextToken at: 2). "Strip the first <u> tag"
	[(scanner peekToken at: 2) = '</u>'] whileFalse:
		[(scanner peekToken at: 2) isNil ifTrue: [self error: 'No
end underline'].
```

```
theHTMLviewer displayWordUnderline: (scanner nextToken at:
2)].
(scanner nextToken at: 2). "Strip the closing </u> tag"! !

!HTMLParser methodsFor: 'HTMLheader' stamp: 'lds 5/9/1999 01:53'!
document

	| htmltag  |
	htmltag := scanner nextToken.

	((htmltag at: 2) = '<html>') ifFalse: [theHTMLviewer display404.
self error: 'HTML tag missing'].
	self header.
	self body.

	htmltag := scanner nextToken.
	((htmltag at: 2) = '</html>') ifFalse: [self error: 'HTML end tag
missing'].! !

!HTMLParser methodsFor: 'HTMLheader' stamp: 'lds 5/9/1999 02:09'!
header

	| headertag  |
	headertag := scanner nextToken.

	[(headertag at: 2) = (Character cr).] whileTrue: [
		Transcript show: 'carriage return'.
		headertag := scanner nextToken.
	].

	((headertag at: 2) = '<head>') ifFalse: [self error: 'Header tag
missing'].
	headertag := scanner nextToken.
	((headertag at: 2) = '<title>') ifFalse: [self error: 'Title tag
missing'].
	self titleString.
	headertag := scanner nextToken.
	((headertag at: 2) = '</title>') ifFalse: [self error: 'Title end
tag missing'].
	headertag := scanner nextToken.
```

```
	((headertag at: 2) = '</head>') ifFalse: [self error: 'Header end
tag missing'].! !

!HTMLParser methodsFor: 'HTMLheader' stamp: 'lds 5/9/1999 01:41'!
titleString

	| titleString |

	titleString := ''.

	[(scanner peekToken at: 2) = '</title>'] whileFalse:

		[(scanner peekToken at: 2) isNil ifTrue: [self error: 'No
end title'].

		titleString := titleString, ' ', (scanner nextToken at:
2)].

	theHTMLviewer displayTitle: titleString.

! !

!HTMLParser methodsFor: 'accessor' stamp: 'lds 5/6/1999 21:55'!
scanner: myScanner

	scanner := myScanner! !

!HTMLParser methodsFor: 'accessor' stamp: 'lds 5/8/1999 14:44'!
talkToHTMLviewer: aHTMLviewer

	theHTMLviewer := aHTMLviewer.
```

```
Object subclass: #HTMLScanner
	instanceVariableNames: 'stream nextCharacter peekedToken value '
	classVariableNames: ''
	poolDictionaries: ''
	category: 'AssignIII'!

!HTMLScanner methodsFor: 'accessing' stamp: 'lds 5/7/1999 13:16'!
stream: aStream

	"Set the stream for reading input from"

	stream := aStream

! !


!HTMLScanner methodsFor: 'scanning' stamp: 'lds 5/7/1999 13:13'!
next

	| returnValue |

	"If a character has been peeked, then return that first."

	nextCharacter isNil

	ifFalse: [returnValue := nextCharacter.

		nextCharacter := nil.

		^returnValue].

	stream atEnd

	ifTrue: [^nil]

	ifFalse: [^stream next]

! !

!HTMLScanner methodsFor: 'scanning' stamp: 'lds 5/7/1999 13:14'!
nextToken

	| return |

	peekedToken isNil ifFalse:

		[return := peekedToken. peekedToken := nil. ^return].

	value := ''. "Empty string"

	^self start

! !

!HTMLScanner methodsFor: 'scanning' stamp: 'lds 5/7/1999 13:14'!
peek

	| returnValue |

	"If a character has been peeked, then return that first."

	nextCharacter isNil

	ifFalse: [returnValue := nextCharacter.

		nextCharacter := nil.

		^returnValue].

	"Otherwise, get a character to peek."

	stream isEmpty

	ifTrue: [nextCharacter := nil]
```

```
        ifFalse: [nextCharacter := stream next].
        ^ nextCharacter
! !
!HTMLScanner methodsFor: 'scanning' stamp: 'lds 5/7/1999 13:14'!
peekToken
        "Transcript show: 'Peeked: ',peekedToken printString."
        peekedToken isNil ifFalse: [^peekedToken].
        peekedToken := self nextToken.
        ^ peekedToken
! !
!HTMLScanner methodsFor: 'scanning' stamp: 'lds 5/7/1999 13:14'!
push: character
        "If a character has been peeked, complain."
        nextCharacter isNil
        ifFalse: [self error: 'already got one in queue!!']
        ifTrue: [nextCharacter := character].! !
!HTMLScanner methodsFor: 'scanning' stamp: 'lds 5/12/1999 23:32'!
start
        | ch |
        ch := self next.
        ch isNil ifTrue: [^Array with: 'start' with: value]. "Nothing
left"
        value := value , (ch asString).
        (ch = Character space or: [ch = Character cr or: [ch = (Character
value: 10)]])
        ifTrue: [value := value allButLast.
                ^self start].
        (ch = (Character value: 60)) "A less-than sign"
        ifTrue: [^self tagRead].
        "Otherwise, we're starting a word"
        ^self startWord
! !
!HTMLScanner methodsFor: 'scanning' stamp: 'lds 5/7/1999 13:15'!
startWord
        | ch |
        ch := self next.
        ch isNil ifTrue: [^Array with: 'startWord' with: value]. "Nothing
left"
        value := value , (ch asString).
        (ch = Character space or: [ch = Character cr])
```

```
        ifTrue: [self push: ch.
                value := value allButLast.
                ^self word ].
        (ch = (Character value: 60)) "A less-than sign"
        ifTrue: [self push: ch.
                value := value allButLast.
                ^self word].
        "Otherwise, we're continuing a word"
        ^self startWord
! !
!HTMLScanner methodsFor: 'scanning' stamp: 'lds 5/7/1999 13:15'!
tag
        ^Array with: 'tag' with: value
! !
!HTMLScanner methodsFor: 'scanning' stamp: 'lds 5/7/1999 13:15'!
tagRead
        | ch |
        ch := self next.
        ch isNil ifTrue: [^Array with: 'tagRead' with: value]. "Nothing
left"
        value := value , ch asString.
        (ch = (Character value: 62)) "A greater-than sign"
        ifTrue: [^self tag].
        "Otherwise, stay in this state"
        ^self tagRead
! !
!HTMLScanner methodsFor: 'scanning' stamp: 'lds 5/7/1999 13:16'!
word
        ^Array with: 'word' with: value
! !
```

```
Object subclass: #HTMLviewer
        instanceVariableNames: 'thePage winheight winwidth penwidth xoffset yoffset fileURL
wordPosition wordoffset linesize spacewidth centeredLine centeredWidth wordStart '
        classVariableNames: ''
        poolDictionaries: ''
        category: 'AssignIII'!

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/8/1999 13:02'!
display404

| fourOhFour |

Transcript show: 'Error'.

fourOhFour := 'Error 404: The page ', (thePage pageURL), ' could not be

loaded.'.

fourOhFour displayAt: wordPosition.! !

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/9/1999 03:11'!
displayBreak

wordPosition := (wordoffset@((wordPosition y) + linesize))! !

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/12/1999 22:33'!
displayCenter: word

| wordBox |

wordBox _ DisplayText text: (word asText) textStyle: TextStyle default

copy centered.

"Transcript show: ('  ', word)."

        wordPosition y < (winheight + yoffset - 20) & (wordPosition y >

yoffset)

                ifTrue:[

(((wordPosition x) + (wordBox form boundingBox width) > winwidth) | (word

= '</center>')) ifTrue: [


        wordPosition := (xoffset + wordoffset@(wordPosition y) +

linesize).

        wordPosition := (((winwidth + xoffset - centeredWidth) quo:

2)@wordPosition y).


        "print out the line"

        centeredLine do: [:aWord | aWord displayAt: (wordPosition).

                wordPosition := wordPosition + (aWord width@0) +

(spacewidth@0)].


        wordPosition := (wordoffset@((wordPosition y) + linesize)).


        "reset the collection of words"

        centeredLine := OrderedCollection new.
```

```
        centeredWidth := 0.
        ]].

(word = '</center>') ifFalse: [
centeredLine add: wordBox.].
centeredWidth := centeredWidth + (wordBox form boundingBox width).
! !

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/10/1999 00:48'!
displayHorizRule


| myPen |


myPen := Pen new.
myPen roundNib: 2.


wordPosition := (wordoffset@((wordPosition y) + linesize)).


((wordPosition y < (winheight + yoffset - 20)) & (wordPosition y >
yoffset)) ifTrue: [
myPen up.
myPen goto: (wordPosition).
myPen north.
myPen turn: 90.
myPen color: (Color black).
myPen down.
myPen go: (winwidth - xoffset - 30).
].


wordPosition := (wordoffset@((wordPosition y) + linesize)).! !

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/9/1999 21:13'!
displayImage: word


| wordBox |


wordBox _ DisplayText text: (word asText) textStyle: TextStyle default
copy centered.


((wordPosition x) + (wordBox form boundingBox width) > winwidth) ifTrue: [
```

```
wordPosition := (wordoffset@((wordPosition y) + linesize))].
word displayAt: (wordPosition).


wordPosition := (wordPosition + ((wordBox form boundingBox width)@0) +
(spacewidth@0)).
! !

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/8/1999 14:40'!
displaySource


"I was using this for testing"
"
        | titleString |

    titleString := ('Viewing : ', ((thePage pageURL) asString), ' asdf',
(thePage sizeof) asString, ' bytes)').
        titleString displayAt: ((xoffset + 20)@(yoffset + 2)).
"

"Transcript show: ((thePage pageURL) printString); cr.
Transcript show: titleString."
"
[Sensor anyButtonPressed] whileFalse:
        [(thePage contentsOf) asString displayAt: ((xoffset + 20)@(yoffset
+ 20 + penwidth))].
"! !

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/13/1999 00:37'!
displayTitle: pageTitle
    | titleString |

    titleString := ('Viewing >> ', pageTitle, ' (',
            (thePage pageURL), ',', (thePage sizeof) asString, ' bytes)').

      titleString displayAt: (xoffset + 20@yoffset + 2).

! !

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/10/1999 00:16'!
displayWord: word
| wordBox |

wordBox _ DisplayText text: (word asText) textStyle: TextStyle default
copy centered.

((wordPosition x) + (wordBox form boundingBox width) > winwidth) ifTrue: [
wordPosition := (wordoffset@((wordPosition y) + linesize))].


((wordPosition y < (winheight + yoffset - 20)) & (wordPosition y >
yoffset)) ifTrue: [
```

```
word displayAt: (wordPosition).

].

"
Transcript show: (wordBox form boundingBox printString) ; cr.

Transcript show: ((wordBox form boundingBox width) printString).

"

wordPosition := (wordPosition + ((wordBox form boundingBox width)@0) +

(spacewidth@0)).

! !

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/10/1999 01:59'!
displayWordBold: word


| wordBox myText |

myText := (word asText).

myText allBold.

wordBox _ (DisplayText text: myText textStyle: TextStyle default copy

centered).


((wordPosition x) + (wordBox form boundingBox width) > winwidth) ifTrue: [

wordPosition := (wordoffset@((wordPosition y) + linesize))].


((wordPosition y < (winheight + yoffset - 20)) & (wordPosition y >

yoffset)) ifTrue: [

wordBox displayAt: (wordPosition).

].

wordPosition := (wordPosition + ((wordBox form boundingBox width)@0) +

(spacewidth@0)).! !

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/10/1999 02:00'!
displayWordH1: word

| wordBox theString myText |

theString := word.

theString := theString asUppercase.


myText := (theString asText) addAttribute: TextEmphasis bold.

myText addAttribute: TextEmphasis underlined.


wordBox _ DisplayText text: myText textStyle: TextStyle default copy

centered.
```

```
((wordPosition x) + (wordBox form boundingBox width) > winwidth) ifTrue: [
wordPosition := (wordoffset@((wordPosition y) + linesize))].

((wordPosition y < (winheight + yoffset)) & (wordPosition y > yoffset))
ifTrue: [
wordBox displayAt: (wordPosition).
].

wordPosition := (wordPosition + ((wordBox form boundingBox width)@0) +
(spacewidth@0)).! !

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/10/1999 02:00'!
displayWordH2: word
| wordBox theString myText |
theString := word.
theString := theString asUppercase.

myText := (theString asText) addAttribute: TextEmphasis bold.
wordBox _ DisplayText text: myText textStyle: TextStyle default copy
centered.

((wordPosition x) + (wordBox form boundingBox width) > winwidth) ifTrue: [
wordPosition := (wordoffset@((wordPosition y) + linesize))].

((wordPosition y < (winheight + yoffset - 20)) & (wordPosition y >
yoffset)) ifTrue: [
wordBox displayAt: (wordPosition).
].

wordPosition := (wordPosition + ((wordBox form boundingBox width)@0) +
(spacewidth@0)).! !

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/10/1999 02:00'!
displayWordH3: word
| wordBox theString |
theString := word.
theString := theString asUppercase.

"
Transcript show: theString.
"
```

```
wordBox _ DisplayText text: (theString asText) textStyle: TextStyle
default copy centered.

((wordPosition x) + (wordBox form boundingBox width) > winwidth) ifTrue: [
wordPosition := (wordoffset@((wordPosition y) + linesize))].

((wordPosition y < (winheight + yoffset - 20)) & (wordPosition y >
yoffset)) ifTrue: [
wordBox displayAt: (wordPosition).
].

wordPosition := (wordPosition + ((wordBox form boundingBox width)@0) +
(spacewidth@0)).! !
```

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/10/1999 02:00'!
displayWordItalic: word

```
| wordBox myText |

myText := (word asText).
myText addAttribute: TextEmphasis italic.

wordBox _ (DisplayText text: myText textStyle: TextStyle default copy
centered).

((wordPosition x) + (wordBox form boundingBox width) > winwidth) ifTrue: [
wordPosition := (wordoffset@((wordPosition y) + linesize))].

((wordPosition y < (winheight + yoffset - 20)) & (wordPosition y >
yoffset)) ifTrue: [
wordBox displayAt: (wordPosition).
].
wordPosition := (wordPosition + ((wordBox form boundingBox width)@0) +
(spacewidth@0)).! !
```

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/10/1999 02:00'!
displayWordUnderline: word

```
| wordBox myText |

myText := (word asText).
myText addAttribute: TextEmphasis underlined.

wordBox _ (DisplayText text: myText textStyle: TextStyle default copy
centered).

((wordPosition x) + (wordBox form boundingBox width) > winwidth) ifTrue: [
wordPosition := (wordoffset@((wordPosition y) + linesize))].

((wordPosition y < (winheight + yoffset - 20)) & (wordPosition y >
yoffset)) ifTrue: [
wordBox displayAt: (wordPosition).
].

wordPosition := (wordPosition + ((wordBox form boundingBox width)@0) +
(spacewidth@0)).! !
```

!HTMLviewer methodsFor: 'display' stamp: 'lds 5/12/1999 22:26'!
makeForm

```
| myPen myForm upButton downButton exitButton |
winwidth := 640.
winheight := 500. "500 pix for display, and 20 pix for top bar"
xoffset := 20.
yoffset := 20.
penwidth := 3.
myForm := (Form extent: winwidth@winheight) fillWhite.
myForm displayAt:(xoffset@yoffset).
myPen := Pen new.
myPen roundNib: penwidth.
myPen up.
myPen goto: (xoffset@(yoffset + winheight)).
myPen north.
myPen color: (Color gray).
```

```
myPen down.
myPen go: winheight.
myPen turn: 90.
myPen go: winwidth.
myPen color: (Color black).
myPen turn: 90.
myPen go: winheight.
myPen turn: 90.
myPen go: winwidth.
myPen up.
myPen color: (Color gray).
myPen goto: (xoffset@(yoffset + 20)).
myPen turn: 180.
myPen down.
myPen go: (winwidth - penwidth).
myPen up.

        upButton _ ViewerButton new.
        downButton _ ViewerButton new.
        exitButton _ ViewerButton new.
        upButton name: 'UP'.
        upButton position: winwidth - 160 + xoffset @ yoffset + 1.
        upButton draw.
        downButton name: 'DOWN'.
        downButton position: winwidth - 120 + xoffset @ yoffset + 1.
        downButton draw.
        exitButton name: 'EXIT'.
        exitButton position: winwidth - 80 + xoffset @ yoffset + 1.
        exitButton draw.

"'UP' displayAt: (winwidth - 120@yoffset + 2).
'DOWN' displayAt: (winwidth - 80@yoffset + 2).
'EXIT' displayAt: (winwidth - 40@yoffset + 2)."
! !

!HTMLviewer methodsFor: 'modifier' stamp: 'lds 5/1/1999 16:42'!
fileURL: theURL
```

```
    fileURL := theURL.! !

!HTMLviewer methodsFor: 'modifier' stamp: 'lds 5/10/1999 00:04'!
wordOffset: offset

wordoffset := offset.! !

!HTMLviewer methodsFor: 'modifier' stamp: 'lds 5/10/1999 00:26'!
wordPosition: position

wordPosition := position.! !

!HTMLviewer methodsFor: 'modifier' stamp: 'lds 5/10/1999 00:39'!
wordStart: position

wordStart := position.! !


!HTMLviewer methodsFor: 'intialize' stamp: 'lds 5/10/1999 00:40'!
initalize: param



      thePage := ThePage new.



      thePage loadPage: param.


      xoffset := 20.


      yoffset := 20.


      wordoffset := 30.


      wordPosition := ((xoffset@yoffset) + (wordStart)).
      linesize := 30.
      spacewidth := 5.
      centeredLine := OrderedCollection new.
      centeredWidth := 0.! !

!HTMLviewer methodsFor: 'accessor' stamp: 'lds 5/10/1999 00:29'!
getPage

^thePage! !

!HTMLviewer methodsFor: 'accessor' stamp: 'lds 5/9/1999 23:53'!
winheight

^winheight! !

!HTMLviewer methodsFor: 'accessor' stamp: 'lds 5/9/1999 23:52'!
winwidth

^winwidth! !

!HTMLviewer methodsFor: 'accessor' stamp: 'lds 5/10/1999 00:53'!
```

```
wordPosition

^wordPosition! !

!HTMLviewer methodsFor: 'accessor' stamp: 'lds 5/10/1999 00:41'!
wordStart

^wordStart! !

!HTMLviewer methodsFor: 'accessor' stamp: 'lds 5/10/1999 00:04'!
wordoffset

^wordoffset! !

!HTMLviewer methodsFor: 'accessor' stamp: 'lds 5/9/1999 23:52'!
xoffset

^xoffset! !

!HTMLviewer methodsFor: 'accessor' stamp: 'lds 5/9/1999 23:52'!
yoffset

^yoffset! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

HTMLviewer class
	instanceVariableNames: 'myViewer myPage oldDisplay '!

!HTMLviewer class methodsFor: 'initalize' stamp: 'lds 5/10/1999 02:55'!
on: param


|  tags parser |


myViewer := HTMLviewer new.

myViewer wordStart: (30@30).

myViewer initalize: param. "sets the page to work with"


oldDisplay := Form fromDisplay: (((myViewer xoffset - 2)@(myViewer yoffset

- 2)) extent: (650@505)).


myViewer makeForm.

myPage := myViewer getPage. "gets the page to work with"

tags := HTMLScanner new.

tags stream: (ReadStream on: (myPage contentsOf) ).

parser := HTMLParser new.

parser scanner: tags.

parser talkToHTMLviewer: myViewer.

parser document.


self control.


! !
```

```
!HTMLviewer class methodsFor: 'control' stamp: 'lds 5/13/1999 00:05'!
control
| xoffset winwidth yoffset keeplooping tags parser thePage thePosition
scrollLength |

scrollLength := 240.

keeplooping := 'true'.
xoffset := myViewer xoffset.
winwidth := myViewer winwidth.
yoffset := myViewer yoffset.
thePage := myViewer getPage.
thePosition := myViewer wordStart.

[Sensor anyButtonPressed] whileTrue:[

((Sensor cursorPoint > (xoffset + winwidth - 120@yoffset)) & (Sensor
cursorPoint < (xoffset + winwidth - 81@yoffset + 20)))

ifTrue: [
        thePosition := (thePosition - (0@scrollLength)).

        myViewer wordOffset: ( 0@100).
myViewer := HTMLviewer new.
myViewer wordStart: (thePosition).
myViewer initalize: (thePage pageURL). "sets the page to work with"
myViewer makeForm.
myPage := myViewer getPage. "gets the page to work with"
tags := HTMLScanner new.
tags stream: (ReadStream on: (myPage contentsOf) ).
parser := HTMLParser new.
parser scanner: tags.
parser talkToHTMLviewer: myViewer.
parser document.
].
```

```
((Sensor cursorPoint > (xoffset + winwidth - 160@yoffset)) & (Sensor
cursorPoint < (xoffset + winwidth - 141@yoffset + 20))

ifTrue: [
        thePosition := (thePosition + (0@scrollLength)).


        myViewer wordOffset: ( 0@100).
myViewer := HTMLviewer new.
myViewer wordStart: (thePosition).
myViewer initalize: (thePage pageURL). "sets the page to work with"
myViewer makeForm.
myPage := myViewer getPage. "gets the page to work with"
tags := HTMLScanner new.
tags stream: (ReadStream on: (myPage contentsOf) ).
parser := HTMLParser new.
parser scanner: tags.
parser talkToHTMLviewer: myViewer.
parser document.

"       Transcript show: 'UP';cr.
        Transcript show: (Sensor cursorPoint asString)."


].

((Sensor cursorPoint > (xoffset + winwidth - 80@yoffset)) & (Sensor
cursorPoint < (xoffset + winwidth - 41@yoffset + 20)))

ifTrue: [
"       Transcript show: 'EXIT';cr.
        Transcript show: (Sensor cursorPoint asString)."
        keeplooping := 'false'.
        oldDisplay displayAt: (myViewer xoffset@yoffset - 2).
].
```

```
].

(keeplooping = 'true') ifTrue:
[self control.]! !
```

```
Object subclass: #ThePage
        instanceVariableNames: 'pageContents pageURL '
        classVariableNames: ''
        poolDictionaries: ''
        category: 'AssignIII'!

!ThePage methodsFor: 'accessors' stamp: 'lds 5/6/1999 22:18'!
contentsOf


 ^pageContents



! !

!ThePage methodsFor: 'accessors' stamp: 'lds 5/8/1999 14:43'!
loadPage: fileURL


     "loads a webpage to work with... takes the URL, loads the

stream/file, then converts it to a      String"


     "fileURL is the URL of the page to be converted to a String"


pageURL := fileURL.


((fileURL copyFrom: 1 to: 4) = 'file') ifTrue: [pageContents :=

(FileStream fileNamed: (fileURL copyFrom: 8 to: (fileURL size)))

contentsOfEntireFile]


ifFalse: [


((fileURL copyFrom: 1 to: 4) = 'http') ifTrue: [pageContents :=

(HTTPSocket httpGet: fileURL) contents]


ifFalse: [pageContents := 'Unsupported Protocol'].
```

```
"Transcript show: (pageContents)."

! !

!ThePage methodsFor: 'accessors' stamp: 'lds 5/6/1999 22:21'!
pageURL



^pageURL! !

!ThePage methodsFor: 'accessors' stamp: 'lds 5/6/1999 22:21'!
sizeof



^ (pageContents size)! !
```

```
Object subclass: #ViewerButton
        instanceVariableNames: 'name position '
        classVariableNames: ''
        poolDictionaries: ''
        category: 'AssignIII'!

!ViewerButton methodsFor: 'init' stamp: 'lds 5/12/1999 22:50'!
draw

        | myPen |

        name displayAt: position + (5 @ 0).

        myPen _ Pen new.

        myPen color: (Color gray).

        myPen up.

        myPen goto: position.

        myPen down.

        myPen north.

        myPen turn: 90.

        myPen go: 40.

        myPen turn: 90.

        myPen go: 20.

        myPen turn: 90.

        myPen go: 40.

        myPen turn: 90.

        myPen go: 20! !

!ViewerButton methodsFor: 'init' stamp: 'lds 5/12/1999 22:22'!
name: theName

        name _ theName! !

!ViewerButton methodsFor: 'init' stamp: 'lds 5/12/1999 22:22'!
position: thePosition

        position _ thePosition! !
```

Assignment 4

Build Assignment 1. See due date on Class Schedule

Discuss on Newsgroup or on http://pbl.cc.gatech.edu/cs2390/1334.html

Everyone has too much information to deal with. Websites, even collaborative ones, newsgroups, email, etc. are overwhelming us. Websites and newsgroups are particularly insidious because they grow without us noticing.

In this assignment, you will define a system to help you monitor activities in each of these.

I expect you to use your system like this:

```
myMonitor := NewsMonitor new.  "Create a newsmonitor"
myMonitor lastUpdate: 3/31/99'. "Show me everything new since 3/31/99"
myMonitor add:
   (NewsgroupChecker for: 'git.cc.class.2390'). "I want to see new things in the git.cc.class.
                                                 2390 newsgroup
myMonitor add:
   (CowebChecker for: 'http://pbl.cc.gatech.edu/cs2390/recent.html'). "I want to se
                                                 what's new since 3/31/99 in
                                                 the cs 2390 coweb"
myMonitor add:
   (WebChecker for: 'http://www.cc.gatech.edu/classes/cs2390_99_spring/'
   limit: 'http://www.cc.gatech.edu/classes/cs2390_99_spring/'
   since: 'webcrawl').
   "Show me all new pages since those recorded in the 2390webcrawl file.
Only follow links that are preceded with 'http://www.cc.gatech.edu/classes/cs2390_99_spring/
Updates the 2390webcrawl file with new crawl data."

myMonitor processTo: 'new.html'. "Do the checking, creating an HTML file with all the new stu
myMonitor processAndShow. "Do the checking, to whatever file the system chooses, then open
                                 Scamper on the result."
```

What appears in the HTML file (and in Scamper) should be a well-formatted display of the news information:

- Links to each web page that changed
- Links to each CoWeb page that changed
- HEADERS-ONLY (titles and authors and dates) of all new newsgroup postings
- Appropriate headers and horizontal rules to make the information more readable.

Scenarios

- Teacher has added a new note to the bottom of a News page, which is linked to the home page. How will you find it and display it?
- There are no new newsgroup postings since the last specified date.

Tips on Implementation

DO NOT USE YOUR DESIGN FROM A1! You've learned things since then. Improve your design.

To read newsgroups, try `SimpleClientSocket nntpTest`. You may want to use some of the PatternTransform projects on the **Cases** (http://pbl.cc.gatech.edu/cs2390/8.html) page to find the date. And remember that the Date class can convert a lot of date formats -- you may not need to do the parsing yourself.

To read webpages, try `HTTPSocket httpShowPage: 'www.altavista.digital.com/'` . To determine if a webpage has changed, use your checksum method from Assignment 2 (Yes, you can improve it.) To open a file in Scamper, try this code:

```
| browser |
browser := Scamper new.
browser jumpToUrl: 'file://',filename.
browser openAsMorph.
```

Grading

DESIGN DOCUMENTATION (50 points) See Design Documentation on what is expected.

- 10 for Diagram: Correct in notation (10), doesn't break any semantic rules (e.g., subclass = is-a) (10).
- 15 points for Good Overall Quality OOA/D: HAS MORE THAN ONE CLASS ON IT, follows rules in Criteria for Good OOA/OOD (e.g., no class is named as a verb) (10). Includes all the classes and services required by the above examples.
- 5 for Good CRC: Makes sense (10); detailed list of collaborators and responsibilities (10).
- 5 for good treatment of Scenarios: Detailed description of how the pieces fit together.
- 5 for Reuse -- makes sense, is believable, doesn't involve redefining anything.

PROGRAM GRADING (50 points)

- 10 Handles CoWeb updates correctly -- that is, gets all changes since (including) the specified date, puts them in a file with valid links.
- 15 Handles newsgroup updates correctly -- that is, gets all new notes since the specified date, puts them all in the HTML file, formats them appropriately.
- 15 Handles webcrawling correctly -- visits all the necessary pages and no others, stores the crawl data in the specified file.
- 5 Handles web page updates correctly -- finds all the pages changed since the last crawl,
- 5 Opens scamper on the output file

Back to the top | Over to CoWeb

24 May 99 Date = Ordered Collection

if (theToken == Date[1] && peekToken == Date[2]

April 27, 1999

NAME: Levi D. Smith

STUDENT NU [redacted] gte187k

10

(25) 1. Guard Station OOA/D

You have been hired to design a guard station and tracking system for a hotel. The idea is for night guards to walk around with personalized keys that they insert into guard stations around the building. The central guard station database logs each guard entry so that it's possible to figure out which sections of the building were guarded at what times. But also, if an invalid key is entered, an alarm should sound to indicate that someone is trying to trick the system. Each key has no processor in it, but can be read by the reader to get a unique key. NOTE: The keys have no processor nor network connection. You can't "talk to" a key.

- Each guard station reader has a connection to a central key database
- A guard has one unique key
- If a guard loses a key, all the current keys should be made invalid and guards are to be given new keys with new numbers.
- When a guard quits, all the current keys should be made invalid, and only the new keys should be recognized.

(10) (a). Name at least two classes that you would need in the design for this system, and two classes that you might consider but would reject. Give your reasons for rejecting those two classes.

Needed Classes:

System Database - to keep track of the keys and valid codes

Guard Station - reads the key cards

Rejected Classes:

Key - because there is only one way interaction with the key. Seems like the key is more of a "user interface". Plus, the key only contains a code, so it would not need a class since the code would be language dependent.

Guard - he really doesn't have anything to do with the system. He only interacts with it.

12

April 27, 1999

NAME: Levi D. Smith

STUDENT NUM! [redacted] te187k

(15) 1 (b) On the back of this sheet or below, or on a separate sheet that you attach to the exam (remember, each problem will be graded separately!), draw the CRC cards for this scenario:

> **A guard gets a card key, takes it to the door, puts it in the station which checks if it's valid, then the guard station log is updated to show that a valid key was entered at the given station at a given time.**

Class: Station	
Responsibility	Collaborators
• Checks for card validity	• System Database
• Read the data from a key	

Class: System Database	
Responsibility	Collaborators
• Verifies if key is valid	
• Reports to Station if key is valid	• Station
• Sound alarm if key is invalid	
• Update log — log?	
• Reads time for the log	
• has a list of valid keys	

Class KeyLog

April 27, 1999

NAME: Levi D. Smith

STUDENT NUMBER: [redacted] gte187k

17

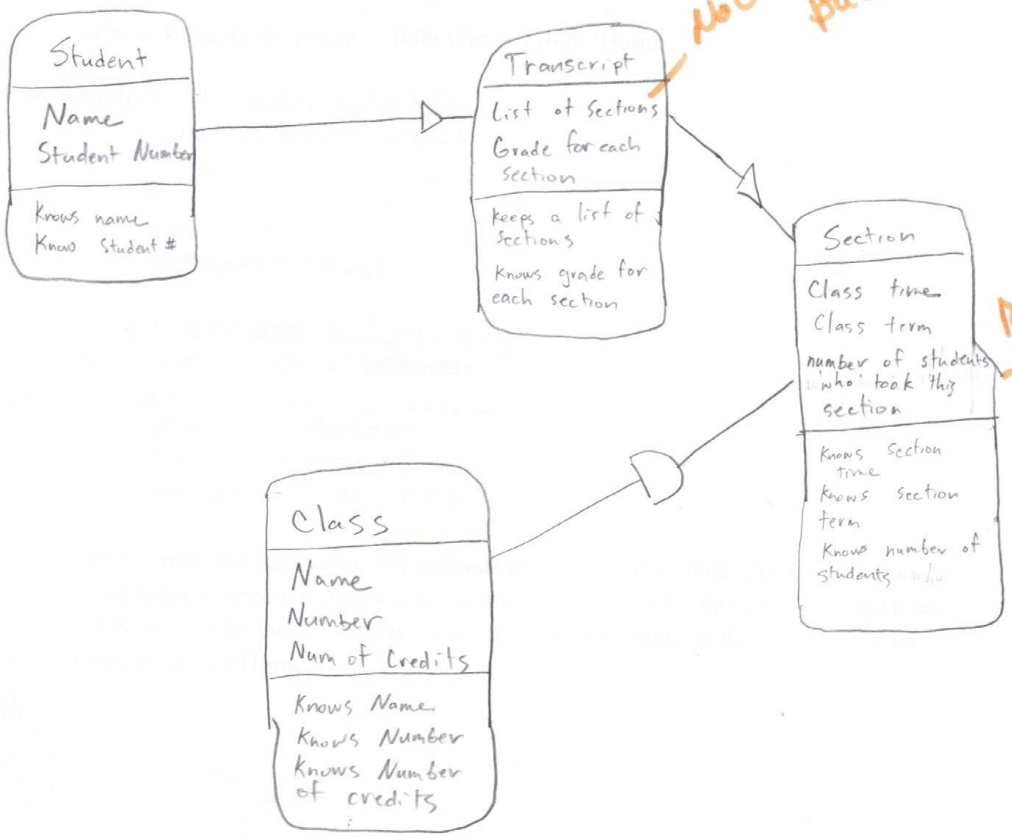
(20) 2. Draw a class diagram

You have just completed an analysis of a student registration system:

- Students have names, student numbers, and a transcript
- Transcripts list the sections that a student has taken and the grade for each.
- A Section is an offering of a Class that has a time and term associated with it, and a number of students who took the class that quarter.
- A Class has a name, a number (e.g. "CS2390"), and a number of credits associated with it.

Draw a class diagram for these classes (below or on back).

Not right notation, but right idea -1



10/1? -2

22/25

April 27, 1999

NAME: Levi D. Smith

STUDENT NUMBE [redacted] gte187t

(35) 3. Implementing Robot Pets

You are building the Smalltalk code for robot pets.

- A RobotPet knows how to bark (for now, print "Bark" to the Transcript), how to wag his tail ("tail wags" to the Transcript), and how to go home.
- When a RobotPet goes home, it sends a message to its RobotHome that it has arrived.
- The RobotHome then begins to recharge the RobotPet (for now, print "Recharging" to the Transcript.)
- RobotPets must know their RobotHomes, and RobotHomes must know their occupants.

(10) (a) Write the definition of **RobotPet** and **RobotHome**

```
Object subclass: #Robot Pet
     instanceVariableNames:
'myRobot Home                                    '
     classVariableNames: ''
     poolDictionaries: ''
     category: 'Robot-Pets'

Object subclass: #Robot Home
     instanceVariableNames:
'occupants                                       '
     classVariableNames: ''
     poolDictionaries: ''
     category: 'Robot-Pets'
```

8/10

(15) (b) Write the **bark**, **wag**, and **goHome** methods for RobotPet. (Assume that there is an **initialize** method which sets up the home of the RobotPet.) Be sure to have **goHome** tell the home who the occupant is, by sending self to the home as the argument for the message **arrivedHome:**

14/15

On Back

bark

"When a Robot Pet barks, a message is sent to the Transcript"

Transcript show: 'bark'.

wag

"When a RobotPet wags his tail, a message is displayed on the Transcript"

Transcript show: 'tail wags'.

goHome

"When a RobotPet goes home, he/she notifies his/her home that they are home.

myRobotHome arrivedHome: self.

April 27, 1999

8

NAME: Levi D. Smith

STUDENT NUMB [redacted] te187k

(10) 3 (c) Write method **arrivedHome**: method (which takes an argument of an occcupant) and the **recharge** methods for RobotHome.

arrivedHome: occupant
"When a RobotPet Arrives home, the home is initialized ?
and the RobotPet is recharged,
~~initialize~~.
Selfrecharge.

---

recharge
"Tells the user that the robot Pet is recharging"

Transcript show: 'Recharging'.

April 27, 1999

NAME: Levi D. Smith

13

STUDENT NUMBER: [redacted] 187A

(20) 4. Short Essay

(2 points free)

-2 (6) (a) What is delegation? What is inheritance? How are they related?

inheritance is delegation where superclass does service for sub class

Delegation is the concept of spreading the job of one object onto the jobs of many objects (abstracting the object).

Inheritance is the ability for one class (subclass) to use and/or modify the methods ~~(messages)~~ of another class (superclass) without having to write all of the code of the superclass over again.

Inheritance is a part of delegation since it allows the superclass to move some of it's responsibilites to the subclasses.

-4 (6) (b) What is the difference between instance methods and class methods? Which object(s) understand each?

Class methods are those methods that are understood by all instances of that class.

Only classes understand/invoke class methods

Instance methods are those understood only by that particular instance of the class.

(6) (c) What are CRC cards and when do you use them in an OOA/D process?

-1

CRC – Class, Responsibility, Collaborator Cards:

CRC Cards help one see how classes interact with each other.

only OOA

They also help one brainstorm ideas in the OOA/D process. (they're cheap)

Classroom SMITH,LEVI CS2390 Spring99
Home View Work

Final-Exam

Grade:

Criteria	Points	Max
p1	15	15
p2	6	10
p3	12	25
p4	10	10
p5	15	15
p6	24	25
Total	82	100

Documentation
Automatically Generated

Work
Automatically Generated

Classroom SMITH,LEVI

Home

Grades

Assignment	Grade	Max	View Turnin
lab			
Lab1	7	10	1 View
Lab2	10	10	1 View
Lab3	10	10	1 View
Lab4	10	10	1 View
Lab5	10	10	1 View
Lab6	9	10	1 View
Lab7	10	10	1 View
project			
Project1	95	100	1 View
Project2	54	100	1 View
Project3	47	105	1 View
Project4	85	90	1 View
Final-Exam	82	100	1 View
			1

ExtraCredit	Ungraded	5	1 View
test			
Midterm	82	100	1 View
FinalExam	Ungraded	100	1 View
FinalGrade	79	100	1 View

0 43100 34026 3