

LCC 6310 The Computer as an Expressive Medium

Lecture 13

Overview

Programming concepts

Processing in Java-mode

Libraries, using other classes/libraries

HTML parsing, intro to HTML parser

Assignment 3 questions?

We started with Processing in...

```
// any code here, no methods  
line(0,0,20,20);
```

We started with Processing in...

```
// any code here, no methods  
line(0 // methods!  
// global vars  
int a;  
// methods  
void setup(){  
}  
void draw(){  
}
```

We started with Processing in...

```
// any code here, no methods
line(0 // methods!
// glo // ...with classes
int a; // (all of the above and then)
// meth
void se class Emotion {
} // fields
void dr // constructor
} // methods
}
```

We started with Processing in...

```
// any code here, no methods
line(0 // methods!
// glo // ...with classes
int a; // (all // ...and subclasses!
// meth class E // (all of the above, and)
void se class Happy extends Emotion {
} // fi // new fields
void dr // co // constructor
} // me // methods
}
```

But in fact...

Everything you write in Processing is actually within a full-up Java class:

```
// Java mode!
class ThatsCrazyTalk extends PApplet {
// setup() and draw() as normally...
// methods
// classes and subclasses
}
```

Java Mode

Processing's Java-mode allows you to program in pure Java

You can import classes that aren't normally imported into a Processing app

Importing means making a classes available to your program - the Java API docs tell you what package classes are in so you know what to import

To go into Java-mode, create a class that extends PApplet

Normally all Processing applets extend PApplet behind the scenes

Take a look at any compiled Processing app

So, those top-level functions that weren't methods of any class really are methods of a class, a class extending PApplet

setup(), draw(), etc. are methods of the class extending PApplet

Template of a Java-mode program

```
class MyProgram extends PApplet {  
    void setup() { ... }  
    void draw() { ... }  
  
    void myTopLevelMethod() { ... }  
  
    class Text { // Text is just an example  
        int xPos, yPos;  
        String word;  
        ...  
    }  
}
```

Notice that any classes you define are **inside** the top class

Why use Java-mode again?

Java-mode gives you access to the entire Java SDK

We need access to some SDK classes for HTML parsing that Processing doesn't make visible by default

Java-mode helps you understand how Processing is built on top of Java

All those "magic" functions and variables are just methods and fields of PApplet that your program inherits

Libraries!

So...what are these libraries and how do I get them?

Libraries are just other classes (in .java or .jar files)

With Java-mode, you can also put your programs in multiple files

Create new tabs (files) with that button in the upper right



Accessing an external class library

Place jars (or class files) within the **libraries** folder in processing

For example, if the class library is called **crazystuff**, create an **crazystuff/library** folder within the libraries folder and add the jar files

Micah will go over how to use external class libraries in your exported applets in one of the tutorial sessions

Use the **import** keyword to bring external classes into your program

Class libraries live in packages

import <packagename>. *; means "make all the classes in <packagename> available for use in my program"

Package names can be hierarchical (e.g. processing.video)

If you get an error that a class can't be found, look in the documentation to see what package you need to import

The imported packages in the example code should get you pretty far

What do you mean parse HTML?

(and why would you want to do it?)

Parsing means to walk through the structure of a file (not just look at it character-by-character, word-by-word)

Look at an HTML file `Results 1 - 20 of about 202 for matrix red ...`

The structure of an HTML file is the tag structure

So parsing means to walk through and interpret the tags

`some text == some text`

If you can parse HTML files, that means you can pull content out of web pages and do stuff with it

Procedural manipulation of web content!

HTML parsers

There are a variety of Java HTML parsers available

proHTML is a Processing library for HTML handling

<http://www.texone.org/prohtml/>

...easy! but poorly documented, and chokes on any malformed HTML...

Java contains HTML parsing capabilities in the swing package

We'll look at this one today

HTML Parser is another free parser that we'll look at later

<http://htmlparser.sourceforge.net/>

There are many others!

There are also other kinds of web services related packages

Switchboard is a web services library for Processing written by IDT graduate Jeff Crouse

Micah will go over the usage of this in one of the tutorials

<http://www.realtimeart.com/switchboard/>

Swing HTML parsing approach

The entry point into the HTML parser is the class `ParserDelegator`

`ParserDelegator` parses an HTML document passed in as a `Reader` and notifies the passed-in `ParserCallback` object as to the state of the parsing

`ParserCallback` implements the following methods

```
public void flush() throws BadLocationException
public void handleText(char[] data, int pos)
public void handleComment(char[] data, int pos)
public void handleStartTag(HTML.Tag t, MutableAttributeSet a, int pos)
public void handleEndTag(HTML.Tag t, int pos)
public void handleSimpleTag(HTML.Tag t, MutableAttributeSet a, int pos)
public void handleError(String errorMsg, int pos)
public void handleEndOfLineString(String eol)
```

The programmer (you!) then creates a `ParserCallback` subclass and fills in these methods to make the parser do what they want

We'll try this out, but first a method overview and some new concepts...

handleText

```
public void handleText(char[] data, int pos)
```

Handles anything that's not a tag (the text between tags)

`data` is an array of characters containing the text

`pos` is the position in the file

handleSimpleTag

```
public void handleSimpleTag(HTML.Tag tag,  
    MutableAttributeSet attrib,  
    int pos)
```

Called for tags like IMG

tag stores the name of the tag

attrib stores any attributes

pos is the position in the file

Example: ``

The tag is **img**

The attributes are **src**, **alt**, **align**, **width** (with their respective values)

handleStartTag

```
public void handleStartTag(HTML.Tag tag,  
    MutableAttributeSet attrib,  
    int pos)
```

Called for tags like BODY

tag stores the name of the tag

attrib stores any attributes

pos is the position in the file

Example: `<body bgcolor="#FFFFFF" topmargin="0" leftmargin="0"
 marginheight="0" marginwidth="0">`

The tag is **body**

The attributes are **bgcolor**, **topmargin**, **leftmargin**, **marginheight**,
marginwidth (with their respective values)

handleEndTag

```
public void handleEndTag(HTML.Tag tag, int pos)
```

Called for tags like ``

tag stores the name of the tag

pos is the position in the file

Exceptions

An **exception** is an event that occurs during the execution of a program that disrupts the normal flow of instructions

Code that might cause certain kinds of exceptions must be enclosed by one of the following:

A **try** statement that catches the exception

```
try {  
    // some code that throws an exception  
} catch (Exception e) {  
    // what to do in case the exception happens  
}
```

A method that specifies that it can **throw** the exception

```
void myNastyMethod() throws Exception {  
    // some code that throws an exception  
}
```

We'll see these in the upcoming examples...

Callbacks

Callback objects are objects that are intended to be passed into another object or method so that they can later be "called back" to do something

Let's look at a simple example...

Here's a callback object that just knows how to print messages:

```
class MsgCallback {
    void printMsg(String msg) {
        println(msg);
    }
}
```

Now let's create an object that will call our MsgCallback object to print a message every second...

Testing the callback...

```
class CallbackTest extends Thread {
    MsgCallback msgcallback = null;
    CallbackTest(MsgCallback cb) {
        msgcallback = cb;
    }
    public void run() { // override the thread run method
        int i = 1;
        while (true) {
            try {
                Thread.sleep(1000);
                msgcallback.printMsg("Message callback number "+i++); // callback
            } catch (InterruptedException e) {
                e.printStackTrace(); // print error in case of exception
            }
        }
    }
}
```

Now we just need to put this in [Processing...](#)

Input streams

Processing provides an `openStream()` method which returns a Java `InputStream` object, so that you can read from a file or a URL

The `InputStream` object can be passed into an `InputStreamReader` object, which acts as a bridge from byte streams to character streams using a specified `Charset` (the default character set is used if none is specified)

The `InputStreamReader` can in turn be passed to a `BufferedReader` which will buffer the characters to make reading more efficient...

```
BufferedReader reader = new BufferedReader(new
    InputStreamReader(openStream("test.html")));
```

Now we're finally ready to move on to the HTML parsing...

Reminder

The entry point into the HTML parser is the class `ParserDelegator`

`ParserDelegator` parses an HTML document passed in as a `Reader` and notifies the passed-in `ParserCallback` object as to the state of the parsing

`ParserCallback` implements the following methods

```
public void flush() throws BadLocationException
public void handleText(char[] data, int pos)
public void handleComment(char[] data, int pos)
public void handleStartTag(HTML.Tag t, MutableAttributeSet a, int pos)
public void handleEndTag(HTML.Tag t, int pos)
public void handleSimpleTag(HTML.Tag t, MutableAttributeSet a, int pos)
public void handleError(String errorMsg, int pos)
public void handleEndOfLineString(String eol)
```

The programmer (you!) then creates a `ParserCallback` subclass and fills in these methods to make the parser do what they want

Let's create a `ParserCallback` subclass...

Subclassing ParserCallback

Check the [API docs](#):

```
javax.swing.text.html.parser.ParserDelegator  
javax.swing.text.html.HTMLEditorKit.ParserCallback
```

A compact way to do this:

```
HTMLEditorKit.ParserCallback callback =  
    new HTMLEditorKit.ParserCallback () {  
        // override the handleText method  
        public void handleText(char[] data, int pos) {  
            System.out.println(data);  
        }  
    };
```

We've subclassed ParserCallback inline with declaring a variable named callback and creating an instance of it!

Let's test it out

Let's try this on a sample [HTML file](#) as follows:

```
try {  
    BufferedReader reader = new BufferedReader(new  
        InputStreamReader(openStream("test.html")));  
    new ParserDelegator().parse(reader, callback, true);  
} catch (IOException e) {  
    e.printStackTrace();  
}
```

Let's see this in [Processing...](#)

How about parsing something online?

All we need to do is open a URL instead of a local File...

```
try {  
    URL url = new URL(href);  
    InputStream instream = url.openStream();  
    Reader reader = new BufferedReader(new InputStreamReader(instream));  
    new ParserDelegator().parse(reader, callback, true);  
    instream.close();  
} catch (IOException e) {  
    e.printStackTrace();  
}
```

Let's see this in [Processing...](#)

And a little more complexity

A few things to do:

- Move our ParserCallback subclass into a separate Java class called MyParserCallback.java

- Parse based on tags and attributes

Let's look at these changes in [Processing...](#)

Assignment 3

Posted online, due **Friday**

- A3-01: Create a subclass of PImage that implements a mosaic(int blockSize) method. The blockSize parameter specifies how big the mosaic block is (e.g. blockSize = 4 would mean the mosaic block size is 4 pixels by 4 pixels). The mosaic method should replace each block of pixels in the image (e.g. if blockSize = 4, each block of 4 by 4 pixels) with the average color value of the pixels in that block. Look at the Pixelate->Mosaic filter in photoshop for an example of what this image operation does. Demonstrate your new class by drawing an image with several different block sizes.
- A3-02: Write a small app that demonstrates kinetic text. Your app should allow the user to type something and move the text around in some way while they type. For example, the user might type text on a line, but slowly the words or letters start drifting apart, or perhaps the line starts bending, or the words and letters flutter to the bottom of the screen, etc. Of course you shouldn't exactly copy any of the typographic examples in Processing or that you find on the web (though using such examples for inspiration, as a place to start modifying code, etc. is fine).

Remember...

For **Thursday** this week: Theory Readings

Two presenters (you know who you are!)

Everyone else: prepare one discussion question for each reading

As We May Think - Vannevar Bush, NMR pp. 35-47

Mythinformation - Langdon Winner, NMR pp. 587-598