CS 121 – Intro to Programming:Java - Lecture 6

Professor Robert Moll (+ TAs)
CS BLDG 276 - 545-4315
moll@cs.umass.edu course web page below

http://twiki-edlab.cs.umass.edu/bin/view/Moll121/WebHome

Announcements

Third OWL assignment up, due next Tuesday; (GET GOING!)

Second Programming assignment due tommorow (9/19). Tip: read 2nd worked example at end of 3.5 VERY CAREFULLY

Midterm posted (W evening 10/15)

TA Office hours: M 12-4; TU 1-3:40; W 4-6; TH 1-4; F 12:30-4:30, in (back room next to) LGRT 223
Java™ 2 Platform Standard Edition 5.0
API Specification

This document is the API specification for the Java 2 Platform Standard Edition 5.0.

See: Description

### Java 2 Platform Packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.applet</td>
<td>Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.</td>
</tr>
<tr>
<td>java.awt</td>
<td>Contains all of the classes for creating user interfaces and for painting graphics and images.</td>
</tr>
<tr>
<td>java.awt.color</td>
<td>Provides classes for color spaces.</td>
</tr>
<tr>
<td>java.awt.datatransfer</td>
<td>Provides interfaces and classes for transferring data between and within applications.</td>
</tr>
<tr>
<td>java.awt.dnd</td>
<td>Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.</td>
</tr>
<tr>
<td>java.awt.event</td>
<td>Provides interfaces and classes for dealing with different types of events fired by AWT components.</td>
</tr>
<tr>
<td>java.awt.font</td>
<td>Provides classes and interface relating to fonts.</td>
</tr>
<tr>
<td>java.awt.geom</td>
<td>Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.</td>
</tr>
<tr>
<td>java.awt.im</td>
<td>Provides classes and interfaces for the input method framework.</td>
</tr>
<tr>
<td>java.awt.im.spi</td>
<td>Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.</td>
</tr>
<tr>
<td>java.awt.image</td>
<td>Provides classes for creating and modifying images.</td>
</tr>
<tr>
<td>java.awt.image.renderable</td>
<td>Provides classes and interfaces for producing rendering-independent images.</td>
</tr>
<tr>
<td>java.awt.print</td>
<td>Provides classes and interfaces for a general printing API.</td>
</tr>
<tr>
<td>java.beans</td>
<td>Contains classes related to developing beans -- components based on the JavaBeans™ architecture.</td>
</tr>
</tbody>
</table>
A package: a bundle of classes with a common general purpose

There are dozens of standard packages in Java

You can make your own packages

When you run standard Java, only the package `java.lang` is automatically loaded (automatically available)

Classes in other packages are loaded as needed

This “load by need” feature makes Java the recycling engine we touted at beginning of term
The Scanner class is in the package java.util. You need to load it in - it’s not automatically available, as the classes in java.lang are.

The import statement loads classes (makes them available to your program):

```java
import java.util.Scanner;
```

Import statements go outside and before your class definitions.
import java.util.Scanner;

public class Adder{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter 2 decimal numbers");
        double num1 = scan.nextDouble();
        double num2 = scan.nextDouble();
        System.out.println("Sum of " + num1 + " " + num2 + " is ");
        System.out.println(num1 + num2);
    }
}
import java.util.*;

public class Paste{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.println("enter first string");
        String s = scan.next();
        System.out.println("enter second string");
        String t = scan.next();
        System.out.println("paste them together");
        System.out.println(s + t);
    }
}

import java.util.Scanner;

public class VerseMaker{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.println("enter animal name");
        String name = scan.next();
        System.out.println("enter animal noise");
        String noise = scan.next();
        MacVerse m = new MacVerse(name,noise);
        m.verse();
    }
}

java VerseMaker

enter animal name (donkey entered)
enter animal noise (hee-haw entered)

and on that farm he had a donkey
ei ei o
With an hee-haw hee-haw here
And a hee-haw hee-haw there
Here a hee-haw there  a hee-haw
Everywhere a hee-haw hee-haw
public class LooseLeaf{
    private int blankPages;
    private String name;

    public LooseLeaf(String who, int blanks){
        blankPages = blanks;
        name = who;
    }
    public int getBlankPages(){
        return blankPages;
    }
    public String getName(){
        return name;
    }
    public void setBlankPages(int amount){
        blankPages = amount;
    }
}

NLP - Natural Language Processing

Text Processing

(Note: many of these slides originated with Prof Andrew McCallum of the CS department)
1967

Stanley Kubrick, filmmaker
1928 - 1999

Arthur C. Clarke, author, futurist,
1917 -
HAL
HAL’s Capabilities

- Display graphics
- Play chess
- *Natural language production and understanding*
- Vision
- Planning
- Learning
- …
HAL Graphics Now
HAL Chess Now
David Bowman: 
Open the pod bay doors, Hal.

HAL: 
I’m sorry, Dave, I’m afraid I can’t do that.

David Bowman: 
What are you talking about, Hal?

...HAL:
I know that you and Frank were planning to disconnect me, and I'm afraid that's something I cannot allow to happen.

Many useful tools, but none that come even close to HAL’s ability to communicate in natural language.
1950
Alan Turing
1912 - 1954

Turing Test

“Computing Machinery and Intelligence”

I propose to consider the question
"Can machines think?"…
We can only see a short distance ahead, but
we can see plenty there that needs to be done.
Layers of Computational Linguistics

1. Phonetics & Phonology
2. Morphology
3. Syntax
4. Semantics
5. Pragmatics
6. Discourse
1. Phonetics & Phonology

The study of: language sounds, how they are physically formed; systems of discrete sounds, e.g. languages’ syllable structure.

**dis-k&-'nekt**

disconnect

“It is easy to recognize speech.”

“It is easy to wreck a nice beach.”

JeetJet?
2. Morphology

The study of the sub-word units of meaning.

**disconnect**

“not”  “to attach”

Even more necessary in some other languages, e.g. Turkish:

*uuygarlastiramadiklarimizdanmissinizcasina*

*uuygar las tir ama dik lar imiz dan mis siniz casina*

(behaving) as if you are among those whom we could not civilize
3. Syntax

The study of the structural relationships between words.

I know that you and Frank were planning to disconnect me.

```
S
  NP       VP
    N      S
      V    SBAR
        SBAR S
          NP VP
            NP CONJ NNP
              you and Frank were planning to disconnect me.
```

Not same structure:

You know me--Frank and I were planning to disconnect that.
4. Semantics

The study of the literal meaning.

I know that you and Frank were planning to disconnect me.

ACTION = disconnect
ACTOR = you and Frank
OBJECT = me
5. Pragmatics

The study of how language is used to accomplish goals.

What should you conclude from the fact I said something? How should you react?

I’m sorry Dave, I’m afraid I can’t do that.

Includes notions of polite and indirect styles.
6. Discourse

The study of linguistic units larger than a single utterance.

The structure of conversations:
- turn taking, thread of meaning.

David Bowman:  
Open the pod bay doors, Hal.

HAL:  
I’m sorry, Dave, I’m afraid I can’t do that.

David Bowman:  
What are you talking about, Hal?

...HAL:  
I know that you and Frank were planning to disconnect me, and I'm afraid that's something I cannot allow to happen.
Linguistic Rules

E.g. Morphology

To make a word plural, add “s”

- dog $\rightarrow$ dogs
- baby $\rightarrow$ babies
- dish $\rightarrow$ dishes
- goose $\rightarrow$ geese
- child $\rightarrow$ children
- fish $\rightarrow$ fish (!)
Inherent ambiguity in syntax

• John didn’t call his wife because he’s mad at her.
Ambiguous Headlines

- Iraqi Head Seeks Arms
- Juvenile Court to Try Shooting Defendant
- Teacher Strikes Idle Kids
- Stolen Painting Found by Tree
- Kids Make Nutritious Snacks
- Local HS Dropouts Cut in Half
- British Left Waffles on Falkland Islands
- Red Tape Holds Up New Bridges
- Clinton Wins on Budget, but More Lies Ahead
- Ban on Nude Dancing on Governor’s Desk
Natural Language Computing is hard because

• Natural language is:
  – highly ambiguous at all levels
  – complex and subtle
  – fuzzy, probabilistic
  – interpretation involves *combining evidence*
  – involves reasoning about the world
  – embedded a social system of people interacting
    • persuading, insulting and amusing them
    • changing over time
Probabilistic Models of Language

To handle this **ambiguity** and to **integrate evidence** from multiple levels we turn to:

The tools of probability:

- Bayesian Classifiers (not rules)
- Hidden Markov Models (not DFAs)
- *Probabilistic* Context Free Grammars

- …other tools of Machine Learning, AI, Statistics
Another Area where Probabilistic Combination of Evidence Won
Natural Language Processing

• Natural Language Processing (NLP) studies how to get computers to do useful things with natural languages:
  – Most commonly Natural Language Understanding
  – The complementary task is Natural Language Generation

• NLP draws on research in Linguistics, Theoretical Computer Science, Artificial Intelligence, Mathematics and Statistics, Psychology, Cognitive Science, etc.
Linguistics

• Linguistics is the study of natural languages:
  – Understanding this naturally-occurring phenomenon.
  – Structure, meaning, how acquired, differences and commonalities across languages.

• Linguistics draws on research in Natural Language Processing, Theoretical Computer Science, Artificial Intelligence, Mathematics and Statistics, Psychology, Cognitive Science, etc.
Some brief history: 1950s

• Early CL on machines less powerful than pocket calculators.
• Foundational work on automata, formal languages, probabilities and information theory.
• First speech systems (Davis et al, Bell Labs).
• MT heavily funded by military, but basically just word substitution programs.
• Little understanding of natural language syntax, semantics, pragmatics.
Some brief history: 1960s

- Alvey report (1966) ends funding for MT in America - the lack of real results realized
- ELIZA (MIT): Fraudulent NLP in a simple pattern matcher psychotherapist
  - It’s true, I am unhappy.
  - *Do you think coming here will make you not to be unhappy?*
  - I need some help; that much is certain.
  - *What would it mean to you if you got some help?*
  - Perhaps I could earn to get along with my mother.
  - *Tell me more about your family.*
- Early corpora: Brown Corpus (Kudera and
Some brief history: 1970s

- Could interpret questions, statements commands.
  - Which cube is sitting on the table?
  - *The large green one which supports the red pyramid.*
  - Is there a large block behind the pyramid?
  - Yes, *three of them. A large red one, a large green cube, and the blue one.*
  - Put a small one onto the green cube with supports a pyramid.
  - OK.
Some brief history: 1980s

- Procedural --> Declarative (including logic programming)
- Separation of processing (parser) from description of linguistic knowledge.
- Representations of meaning: procedural semantics (SHRDLU), semantic nets (Schank), logic (perceived as answer; finally applicable to real languages (Montague)
- Perceived need for KR (Lenat and Cyc)
- Working MT in limited domains (METEO)
Some brief history: 1990s

- Resurgence of finite-state methods for NLP: in practice they are incredibly effective.
- Speech recognition becomes widely usable.
- Large amounts of digital text become widely available and reorient the field. The Web.
- Resurgence of probabilistic / statistical methods, led by a few centers, especially IBM (speech, parsing, Candide MT system), often replacing logic for reasoning.
- Recognition of *ambiguity* as key problem.
- Emphasis on machine learning methods.
Some brief history: 2000s

• A bit early to tell! But maybe:
  – Continued surge in probability, Bayesian methods of evidence combination, and joint inference.
  – Emphasis on meaning and knowledge representation.
  – Emphasis on discourse and dialog.
  – Strong integration of techniques, and levels: bringing together statistical NLP and sophisticated linguistic representations.
  – Increased emphasis on unsupervised learning.
Example Applications of NLP

Natural Language Processing
Natural Language Processing should make it possible for people to use computers in much the same way that they would use a human assistant to get their work ... research.microsoft.com/nlp/ - 28k - Cached - Similar pages

ISI's Natural Language Group
Overview of Research Environment Natural Language Processing at USC/ISI. ... USC offers a wide range of courses in areas related to natural language processing ... Description: The Natural Language Processing group at the Information Sciences Institute of the University of Southern...
Category: Computers > Artificial Intelligence > ... > Research Groups
www.isi.edu/natural-language/nlp-at-isli.html - 15k - Cached - Similar pages

Foundations of Statistical Natural Language Processing
Foundations of Statistical Natural Language Processing. ... Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing, MIT Press. ...
nlp.stanford.edu/fsnlp/ - 7k - Cached - Similar pages

Yahoo! Directory Artificial Intelligence > Natural Language ... Artificial Intelligence > Natural Language Processing Directory > Science > Computer Science > Artificial Intelligence > Natural Language Processing. ...
dir.yahoo.com/Science/Computer_Science/Artificial_Intelligence/ Natural_Language_Processing/ - 19k - Cached - Similar pages

Columbia Natural Language Processing Group
Welcome to the Columbia Natural Language Processing Group home page! To read about our group and our research interests, view our site (Overview...
Example Applications of NLP: MSWord spelling correction, grammar checking
Example Applications of NLP:
News categorization and summarization
Example Applications of NLP

Information Extraction: Find experts, employees

http://www.eliyon.com
Example Applications of NLP: Information Extraction - Job Openings

foodscience.com-Job2
JobTitle: Ice Cream Guru
Employer: foodscience.com
JobCategory: Travel/Hospitality
JobFunction: Food Services
JobLocation: Upper Midwest
Contact Phone: 800-488-2611
DateExtracted: January 8, 2001
Source: www.foodscience.com/jobs_midwest.html

OtherCompanyJobs: foodscience.com-Job1
Example Applications of NLP: Information Extraction: Job Openings
Example Applications of NLP: Automatically Solving Crossword Puzzles
Example Applications of NLP: Question Answering

Question:
who is married to bill gates

Possible answers: XML TXT

- Bill was married to Melinda French Gates in 1994 in Hawaii.
- Mary Gates, Bill's mother, biggest fan, and strongest prodder, finally laid down an ultimatum in 1993. She was dying of cancer, and wanted to see her only son married.
- Bill Gates married Melinda French in Hawaii on January 1, 1994, and his mother died a few months later.
- 1994 Bill Gates and Melinda French married in Hawaii on New Years Day.

Try your question on other engines:
Alta Vista | CNN News Engine | Ask Jeeves | Excite | Google | HotBot | Lycos | Start | Yahoo
Example Applications of NLP:

Machine Translation

Assault or preventive strike? The German attack on the Soviet Union on 22 June 1941.

of Gerhard Baumfalk

Used & again starting from EUR 10.50
Offerer dispatches in 1-2 working-days.
Example Applications of NLP: Automatically generate Harlequin Romance novels?
Text Processing

Tools, and the notion of a “language”
A Language

Some sentences in the language
• The man took the book.  From [Chomsky, 1956], his first context-free parse tree.
• The purple giraffe hopped through the clouds.
• This sentence is false.

Some sentences not in the language
• *The girl, the sidewalk, the chalk, drew.
• *Backwards is sentence this.
• *IoDvaD tlhIngan Hol ghojmoH be.
How can we capture what does and does not belong to a “language”?

Java - actually pretty easy (compiler!)

Natural languages, e.g., English - much, much harder -- and the nature of the capture matters immensely to linguists
Compact description of a language - via a grammar

- The language of legal strings of matched parentheses, for example..
- Start with some “non-terminal” symbol, $S$. 
- Expand that symbol, using some substitution rules.
- ...keep applying rules until all non-terminals are expanded to terminals.
- The string of terminals is in the sentence.
Matched Parentheses

- S -> ()
- S -> (S)
- S -> SS

- So: ((())()) -- a legal string in MP
- S -> (S) -> (SS) -> ((S)) -> ((S)) -> ((SS)) ->
- (((()S)) -> (((())())))
Noam Chomsky
1928 -

Chomsky Hierarchy
Generative Grammar
Libertarian-Socialist

The most cited person alive.
Chomsky Hierarchy

- Type 0 languages (Turing-equivalent)
  Rewrite rules $a \rightarrow b$
  where $a, b$ are any string of terminals and non-terminals

- Context-sensitive languages
  Rewrite rules $aXb \rightarrow acb$
  where $X$ is non-terminal and $a,b$ as above

- Context-free languages
  Rewrite rules $X \rightarrow a$
  where $X, a, b$ as above

- Regular languages
  Rewrite rules $X \rightarrow aY$
  where $X, Y$ are non-terminals and $a$ is a string of terminals

Linguistic example:

- ATNs
- TAGs
- PSGs
- FSAs
Regular language example

- Non-terminals:
  - $S, X, Y, Z$
- Terminals:
  - $m, o$
- Rules:
  - $S \rightarrow mX$
  - $X \rightarrow oY$
  - $Y \rightarrow oY$
  - $Y \rightarrow$
- Start symbol:
  - $S$

An expansion:

- $S$
- $mX$
- $moY$
- $mooY$
- $mooo$
Example: Sheep Language

Strings in and out of the example Regular Language:

• In the language:
  “ba!”, “baa!”, “baaaaa!”
• Not in the language:
  “ba”, “b!”, “ab!”, “bbaaa!”, “alibaba!”

Finite-state Automata

Regular Expression

```
baa*!
```
Recognizer

• A recognizer for a language is a program that takes as input a string $W$ and answers “yes” if $W$ is a sentence in the language, and answers “no” otherwise.

• We can think of this as a machine that emits only two possible responses it input.
The real beauty of regular languages

1) They are immensely more powerful than they appear

2) You don’t need to use a recognizer explicitly - there’s a simple notation (called the notation of “regular expressions”) which exactly captures such languages

3) This notation is amenable to shorthand operators
Regular Languages: related concepts

Regular Languages
the accepted strings

Finite-state Automata
machinery for accepting

Regular Expressions
a way to type the automata
# Transition Table, $\delta$

<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>(b)</th>
<th>(a)</th>
<th>(!)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>1</td>
<td>$\emptyset$</td>
<td>2</td>
<td>$\emptyset$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\emptyset$</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
</tr>
</tbody>
</table>
## Regular Expressions

The "foundational" operations

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Concatenation</strong></td>
<td></td>
</tr>
<tr>
<td>( abc )</td>
<td>( abc )</td>
</tr>
<tr>
<td><strong>Disjunction</strong></td>
<td></td>
</tr>
<tr>
<td>( a \mid b )</td>
<td>( a \mid b )</td>
</tr>
<tr>
<td>( (a \mid bb) d )</td>
<td>( ad \mid bbd )</td>
</tr>
<tr>
<td><strong>Kleene star</strong></td>
<td></td>
</tr>
<tr>
<td>( a^* )</td>
<td>( \varepsilon \mid a \mid aa \mid aaa \mid \ldots )</td>
</tr>
<tr>
<td>( c(a \mid bb)^* )</td>
<td>( ca \mid cbba )</td>
</tr>
</tbody>
</table>

Regular expressions / Finite-state automata are "closed under these operations"
Stephen Kleene, 1909 - 1994

Attended Amherst College!

Best known for founding the branch of mathematical logic known as recursion theory, together with Alonzo Church, Kurt Godel, Alan Turing and others; and for inventing regular expressions.

“Kleeneliness is next to Godeliness.”
Practical Applications of RegEx’s

- Web search
- Word processing, find, substitute
- Validate fields in a database (dates, email addr, URLs)
- Searching corpus for linguistic patterns
  - and gathering stats...

- Finite state machines extensively used for
  - acoustic modeling in speech recognition
  - information extraction (e.g. people & company names)
  - morphology
  - ...
The Gutenberg Project

http://www.gutenberg.org/wiki/Main_Page

Project Gutenberg is the first and largest single collection of free electronic books, or eBooks. Michael Hart, founder of Project Gutenberg, invented eBooks in 1971 and continues to inspire the creation of eBooks and related technologies today.

Last 7 days - 552,812 books have been downloaded
A note on storage

One page of text = 1 KB; 4 books = 1 MB

(example: Heart of Darkness (Conrad): 232K
4000 books = 1 GB

250 GB costs ~ $100 - this is ~ one million books, text only

So it costs $100 to store the UMass library (text only) on your computer!
Two types of characters in REs

- **Literal**
  - Every normal text character is an RE, and denotes itself.

- **Meta-characters**
  - Special characters that allow you to combine REs in various ways
    - Example:
      - \( a \) denotes \( a \)
      - \( a^* \) denotes \( \varepsilon \) or \( a \) or \( aa \) or \( aaa \) or ...
Regular expressions have become especially useful, with the addition of special “short-hand” characters/symbols.

These additions don’t add to the power of the language.

They merely stand as shorthand for humans.

The computer handles their “expansion” behind the scenes.
### Basic Regular Expressions

<table>
<thead>
<tr>
<th>Character Concat</th>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>went</td>
<td>went</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alternatives</th>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>(go</td>
<td>went)</td>
<td>go   went</td>
</tr>
<tr>
<td>[aeiou]</td>
<td>a o u</td>
<td></td>
</tr>
<tr>
<td>[^aeiou]</td>
<td>b c d f g</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>a z &amp;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Loops &amp; skips</th>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>a*</td>
<td>ε a aa aaa ...</td>
<td></td>
</tr>
<tr>
<td>a+</td>
<td>a aa aaa</td>
<td></td>
</tr>
<tr>
<td>colou?r</td>
<td>color colour</td>
<td></td>
</tr>
</tbody>
</table>