Today’s topics

Algorithms
   Java
   Syntax and Grammars

Upcoming
   More Java

Reading
   Great Ideas, Chapter 2
Java!

- Java is a buzzword-enabled language
- From Sun (the developers of Java),
  "Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and dynamic language."

- What do all of those terms mean?
“Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and dynamic language.”

- **A programming language**
  - A vocabulary and set of syntactical (grammatical) rules for instructing a computer to perform specific tasks
  - You can do most anything in any programming language
  - A particular language encourages one to do things in a certain way

- **A Question for the course: What makes a good language?**
“Java is a **simple**, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and dynamic language.”

- Based on popular languages called C and C++
- C: old, pretty bare bones language
- C++: newer, more complicated language
- Start from C and add some of C++’s more useful features
  - From Gosling, the creator, “Java omits many rarely used, poorly understood, confusing features of C++ that in our experience bring more grief than benefits.”

- Question: Is Java really all that simple?
“Java is a simple, **object-oriented**, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and dynamic language.”

- **The object-oriented paradigm**
  - Problems and their solutions are packaged in terms of **classes**
  - The information in a class is the **data**
  - The functionality in a class is the **method**
  - A class provides the framework for building **objects**

- **Object-oriented programming (OOP) allows pieces of programs to be used in other contexts more easily**
A distributed system is one where multiple separate computer systems are involved. Examples include:

- Electronic card catalogs
- The web

Java was designed for the web. How many distributed tasks can you think of that you interact with daily?
“Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and dynamic language.”

- Java a high-level language
- High-level languages must be translated to a computer’s native tongue, machine language
- Interpreted high-level languages are translated to an intermediate form and then converted to machine language and run

- Why?
- We’ll learn more about this later
“Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and dynamic language.”

- Programs will have errors, but a good program degrades reasonably
- A robust program may not do exactly what it is supposed to do, but it should not bring down other unrelated programs down with it

- Question: Give me an example of a non-robust program you have seen?
“Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and dynamic language.”

- **Security**: techniques that ensure that data stored on a computer cannot be read or compromised.
- **A program is running on your computer. What is to stop it from erasing all of your data, accidentally or otherwise?**

- **Question**: Is Java really all that secure?
A language is architecture-neutral if it does not prefer a particular type of computer architectures. For example, the Macintosh processor family (PowerPC) and the PC (x86-Pentium) family have their own respective strengths and weaknesses. It is not too hard to construct a program that will run faster on one than an other.

A particular program is never entirely architecture neutral though.

Question: When is being architecturally neutral a bad thing?
“Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, **portable**, high performance, multi-threaded, and dynamic language.”

- A program is portable if it will work the same (roughly) on many different computer systems
- HTML is also platform-independent or portable
- A whole lot of effort is currently spent *porting* non-portable code
“Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and dynamic language.”

- Performance: speed in completing some task
- Performance is everything to most computer and software manufacturers.

- Story:
  - If the transportation industry kept up with the computer industry, one would be able to now buy a Rolls Royce that could drive across country in 5 minutes for $35.

- Rebuttal:
  - It would crash once a week, killing everyone on board.
“Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, *multi-threaded*, and dynamic language.”

- A thread is a part of the program that can operate independently of its other parts
- Multi-threaded programs can do multiple things at once
  - e.g. download a file from the web while still looking at other web pages

- **Question:** What is the problem with multiple agents working at the same time?
  - Synchronization
“Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and **dynamic** language.”

- Dynamic refers to actions that take place at the moment they are needed rather than in advance
  - Antonym: static

- A dynamic program can
  - Ask for more or less resources as it runs
  - Use the most recent version of some code that is available

- Question: Why is being dynamic a good thing?
import java.awt.*;

public class HelloWorld extends java.applet.Applet
{
    TextField m1;
    public void init()
    {
        m1 = new TextField(60);
        m1.SetText("Hello World");
        add(m1);
    }
}
Definitions

- **Algorithm**: ordered set of unambiguous executable steps, defining a terminating process
- **Program**: instructions executed by a computer
- **Applet**: Java program that is executed in a program such as appletviewer or a Java-enabled web browser
- **Class**: family of components sharing common characteristics consisting of:
  - **Data**: information
  - **Method**: functionality
- **Object**: instance of a class
- **Variable**: represent value stored in computer memory. A variable must be defined or declared before being used
  - Sometimes synonymous with object
Grammar

- English and other natural languages have structure

\[<S> \rightarrow <\text{NOUN-PHRASE}> <\text{VERB-PHRASE}>\]
\[\rightarrow <\text{NOUN}> \mid <\text{ARTICLE}> <\text{NOUN}> \mid <\text{PP}>\]
\[<\text{VERB-PHRASE}> \rightarrow <\text{VERB}> \mid <\text{VERB}> <\text{NOUN-PHRASE}>\]
\[<\text{NOUN}> \Rightarrow \text{DOG} \mid \text{FLEAS} \mid \text{PERSON} \mid \ldots\]
\[<\text{VERB}> \Rightarrow \text{RAN} \mid \text{BIT} \mid \ldots\]

- Process of taking sentence and fitting it to grammar is called *parsing*

DOG BIT PERSON

- Parsing English is complex because of *context dependence*
Formal specifications

- Need a precise notation of syntax of a language
- Grammars can be used for generation and also can be used
- Context-free grammars
  
  \[ \text{name} \rightarrow \text{sequence of letters and/or digits that begins with a letter} \]
  
  \[ \text{name} \rightarrow \text{guessB} \]
  
  \[ \text{name} \rightarrow \text{msg42} \]

- Substitute as many times as necessary. All legal statements can be generated this way
  - Want `person = firstn + " " + lastn;`
  - How do we get this from our grammar?
Random Sentence Generator

- Constructs sentences, paragraphs, and even papers that fit a prescribed format.
- The format is specified by a set of rules called a grammar.
- A grammar consists of a set of definitions.
- Each definition is a set of productions.
- Examples of grammars
  - Extension request
  - College rejection
  - Poem
  - [http://www.cs.duke.edu/courses/fall02/cps001/code/grammars/](http://www.cs.duke.edu/courses/fall02/cps001/code/grammars/)
- Natural languages have grammars

\[ <S> \Rightarrow <NP> <VP> \]
Poem Grammar

- All grammars begin with start rule
  ```
  { 
  <start>
  The <object> <verb> tonight. ; 
  }
  ```

- Nonterminals are indicated by angle brackets
  ```
  { 
  <object>
  waves ;
  big yellow flowers ; 
  slugs ; 
  }
  ```
More on the poem grammar

- **Nonterminals can refer to other nonterminals**

```latex
\{ \\
  \texttt{<verb>} \\
  \texttt{sigh <adverb> ;} \\
  \texttt{portend like <object> ;} \\
\} \\
\{ \\
  \texttt{<adverb>} \\
  \texttt{warily ;} \\
  \texttt{grumpily and <adverb> ;} \\
\}
```
Generating a poem

→ all sentences start with <start>

<start>

→ There is only one production in the definition of <start>

The <object> <verb> tonight.

→ Expand each grammar element from left to right
→ "The" is a terminal, so it is simply printed –
→ <object> is a non-terminal, so it must be expanded

→ Choose one:
  • waves
  • big yellow flowers
  • slugs

→ Suppose that 'slugs' is chosen
Generating a poem

The slugs <verb> tonight.
   -<verb> is a non-terminal, so it must be expanded –
   Choose one:
      1. sigh <adverb>
      2. portend like <object>

The slugs sigh <adverb> tonight.
   <adverb> is a non-terminal, so it must be expanded
      1. warily
      2. grumpily

The slugs sigh grumpily tonight.
   "tonight." is a non-terminal so it is simply printed
   There are no more non-terminals to expand!
   The grammar has generated a complete poem