Today’s topics

Java
- Syntax and Grammars
- Sample Programs

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

- A distributed system is one where multiple separate computer systems are involved
  - Electronic card catalogs
  - The web

- Java was designed for the web
- Question: What are examples of a distributed task in your lives?
“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 carried out (run or executed) using an interpreter.
- 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.

E.g. 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 Roll 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
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?

“Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and dynamic language.”
A First Java Program

```java
import java.awt.*;
public class HelloWorld extends java.applet.Applet {
    TextField m1, m2;
    public void init()
    {
        m1 = new TextField(60);
        m2 = new TextField(60);
        m1.setText("Hello World");
        m2.setText("This is a simple Java test.");
        add(m1);
        add(m2);
    }
}
```
Things to note:

- Program is a class
- Class contains *data* and *methods*
  - Methods also called *functions*
- Method `init()` always started for applets
- `add` statements needed for *layout*
- Applet invoked through HTML file
- Program tested with Web browser or *appletviewer*
  - We will normally use our web pages
- Note points of grammar ...
  - Semicolons ;, braces { }, parentheses ( ), etc.
Sample file

❖ Can have separate web page:

```html
<HTML>
  <HEAD>
    <TITLE> The textfield demo program. </TITLE>
  </HEAD>
  <BODY>
    This tests the textfield capability.
    <APPLET code="HelloWorld.class" WIDTH=750 HEIGHT=325>
    </APPLET>
  </BODY>
</HTML>

❖ Or can incorporate the following line in any web page:

```html
<APPLET code="HelloWorld.class" WIDTH=750 HEIGHT=325>
</APPLET>
```
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 the `appletviewer` or a Java-enabled `web browser`
- **Class**: family of components sharing common characteristics consisting of:
  - **Data**: information
  - **Methods**: 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`
Reflect on our progress

- **What good is HelloWorld?**
  - What have we accomplished?
  - Can link to our web page.
- **Want something more.**
  - Programs should do something for us.
  - Just putting a message on the screen is pretty lame …
- **Program results need to change or vary as a result of:**
  1. Our actions
  2. Other outside data
Decision trees

- **if statements**
  
  ```
  if (logical expression) {
    "true" actions
  }
  ```

- **if-else statements**
  
  ```
  if (logical expression) {
    "true" actions
  }
  else (logical expression 2) {
    "false" actions
  }
  ```

- **Logical expressions**
  
  - analogous to yes or no questions
  - true or false
  
  - **Statements that are true**
    
    - `(5 < 7)`
    - `(100 == 100)`
    - `(100 != 10)`
    - `(10 <= 10)`

  - **Statements that are false**
    
    - `(-2 > -1)`
    - `(10 != 10)`
Using Buttons with if statements

- What does it mean to have an interactive program?
  - Computer must be *waiting* for your actions.
  - Like waiting for the phone to ring for an important call
  - Need something called a “listener”

- Also need to create Buttons
  - Example will show how

- With multiple Buttons, need to know *which one* was pressed
  - Like having different tones for front and back door bell buttons
Program using Buttons: 1

```java
public class TrafficLight extends Applet implements ActionListener {

    TextField m1, m2;
    Button b1, b2, b3;
    public void init ()
    {
        m1 = new TextField(80);
        m1.setText("What are you going to do when the light is:");
        b1 = new Button("GREEN");
        b2 = new Button("YELLOW");
        b3 = new Button("RED");
        m2 = new TextField(80);
        add(m1); add(b1); add(b2); add(b3); add(m2);
    }
}
```
Program using Buttons: 2

```java
b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
```

- Have Invoked the listener with statements above
  - We have “told” the listener about each of the 3 buttons
- **Now: Need to write the listener**
  - Listener must be named `actionPerformed`
  - Using `if` statements, it will figure out which button was pushed and take the desired action
public void actionPerformed(ActionEvent event) {
    Object cause = event.getSource();
    if (cause == b1) {
        m2.setText("Keep on rolling.");
    }
    if (cause == b2) {
        m2.setText("Step on it! You can make it!");
    }
    if (cause == b3) {
        m2.setText("I suppose you'll have to stop.");
    }
}
A decision tree

Would you like to read about a scientist?

0

Would you like to read about Einstein?

1

He received the Physics Prize in 1921.

3

Try the Medicine Prize in 1962.

4

Would you prefer a humanitarian?

2

Look up the Peace Prize in 1991.

5

Try A. Solzhenitsyn, Literature 1970.

6
More Java Syntax

- **Assignment statement**
  
  \[ \text{variable} = \text{expression}; \]

- **Method invocation**
  
  - Also called function or procedure
  - Invoking also called “calling” a function
  - Methods can take *arguments*
  
  ```java
  button.setText("This text is an argument");
  init();
  ```

- **Variable declaration**
  
  ```java
  VariableType variableName;
  Button choice;
  ```