

On to Java!

Corky Cartwright
Stephen Wong
Department of Computer Science
Rice University



From Scheme to Java

- Scheme and Java look completely different
- Don't be fooled. Java is very Scheme-like underneath (perhaps excessively so).
 - Self-identifying data
 - Implicit sharing of objects (discouraging mutation); assignment does not copy!
- C++ → Java?
 - Formerly the Rice curriculum.
 - In industry. Java/C# is dominant. Anachronisms in the JVM have blunted Java dominance somewhat.
- DrScheme → DrJava



Java Notation

- Lots of warts thanks to C/C++ syntax. After an immigration period, they become only minor annoyances.
- What is a Java program? A collection of classes.
- What is a class? Rough answer: a Scheme struct on steroids. Instead of writing functions that manipulate structs, you add "methods" to a class. The methods are attached to each object in the class so they can directly refer to members (fields in Scheme terminology) of the class.
- All Java code belongs to some class.



Guiding Vision

- Program design in Java is data-directed.
 Design the data abstractions first; they will determine the structure of the code. In OOP circles, this data design process is often called object-modeling.
- Software development is incremental and testdriven. Essentially the same design recipe as we used for Functional Programming (FP).
- Key to OO approach: common data and programming abstractions are codified as design patterns (much like templates in FP).



Secondary Theme: DrJava

- DrJava, our lightweight, reactive environment for Java, was created specifically to foster learning to program in Java.
- DrJava facilitates active learning; with DrJava learning Java is a form of exploration.
- DrJava is not a toy; DrJava is developed using DrJava. It includes everything that we believe is important and nothing more.



What Is an Object?

- Collection of *fields* representing the properties of a conceptual or physical object.
- Collection of operations called methods for observing and changing the fields of the object.

These fields and methods often called the *members* of the object.

How Are Objects Defined?

- All objects are created using templates (cookie cutters) just like Scheme structs. (Not the same notion of template as in FP design.)
- Instead of writing define-struct statements, we write class definitions.
- Since all code is contained within a class, class definitions tend to be much richer (and more complex in real world examples) that define-struct statements. After all, the code that would be written in function definitions in Scheme must be written as methods of some class.



Example: a Phone Directory

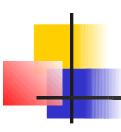
- Task: maintain a directory containing the office address and phone number for each person in the Rice Computer Science Dept.
- Each entry in such a directory has a natural representation as an object with three fields containing a person's
 - name
 - address
 - phone number

represented as character strings.



Summary of Entry Data

- Fields:
 - String name
 - String address
 - String phone
- Accessed only through implicitly generated methods:
 - String name()
 - String address()
 - String phone()



Entry Demo in DrJava

- Write DrJava class code
- Create an object
- How do we perform any computation with it?



Java Method Invocation

A Java method m is executed by sending a method invocation (method call)
 o.m()

to an object o, called the *receiver*. The method m must be a *member* of o.

The code defining the method m can refer to the entire receiver object using the keyword this.



Method Invocation Demo

- Apply some auto-generated methods to an Entry
- How do we build up expressions from method invocations?
 - Apply operators (built-in to Java) on primitive types (int, double, boolean)
 - Invoke methods



Java Expressions

- Java supports essentially the same expressions over primitive types (int, double, boolean) as C/C++.
- Notable differences:
 - boolean is a distinct type from int
 - no unsigned version of integer types
 - explicit long type



Defining (Instance) Methods

- Recall our definition of the Entry class. How can we add methods to this class?
- Suppose we want Entry to support a method:

```
boolean match(String keyname)
invoked by syntax like
e.match("Corky")
```



Method Definition Demo

- Method syntax is C-like.
- Comment notation:
 - // opens a line comment (like ";" in Scheme)
 - Block comments are enclosed in /* ... */

Code for Entry with match

```
class Entry {
  /* fields */
  String name;
  String address;
  String phone;
  /** @return true iff name matches keyName.*/
  boolean match(String keyName) {
    return keyName.equals(name));
```



For Next Class

- Start thinking about the exam
- Optional Homework due Wednesday after break.
- Labs introducing Java tomorrow.
- Reading: OO Design Notes, Ch 1.1 1.4.2.
- Please send me comments on typos and suggestions for improving the notes.